KNOWLEDGE BASED REQUIREMENTS SPECIFICATION FOR RECONFIGURABLE ASSEMBLY SYSTEMS

by Hitendra J Hirani, BSc

The University of Nottingham

Thesis submitted to The University of Nottingham for the degree of Doctor of Philosophy,

February 2005

ABSTRACT

Automated assembly technology may be the key to sustaining manufacturing industry in more developed countries. Currently this comprises dedicated systems that can assemble single products at high volumes and flexible systems to assemble a wide variety of products in low volumes. However, competitive forces demand a compromise between the two and Reconfigurable Assembly Systems are an avenue for achieving high volume and high variety production.

Although this technology is coming to the fore, there is a distinct lack of tools and methods that make the prospect attractive to key decision makers in organisations. Reconfigurable solutions, which may be profitable in the long term, are rejected in favour of short term solutions, which prove to be more expensive over time.

The benefits of requirements engineering have been exploited in software engineering and this work demonstrates how these can be adapted to an assembly environment to form a new basis for communication between the system vendors, who supply assembly system solutions, and system users, who use them.

Knowledge Engineering has become a key aspect in industry due to the challenges of retaining personnel and their knowledge within organisations. This is because employees take their knowledge of the organisation with them when they leave. The retention of this knowledge would help to maintain the continuity within organisations.

This thesis reports on research that aims to provide a means to integrate these three aspects to form a basis for sustaining competitive manufacture in more developed countries.

Moreover, Knowledge Based Requirements Specification for Reconfigurable Assembly Systems will provide a vital medium for promoting Reconfigurable Assembly Systems and encourage their implementation by providing a knowledge-based platform for the specification of Reconfigurable Assembly Systems.

TABLE OF CONTENTS

CHAPTER INDEX

INDEX OF FIGURES

INDEX OF TABLES

LIST OF APPENDICES

GLOSSARY

Below is a definition of terms used in the thesis.

Assembly – the action of assembling component parts or a unit consisting of component parts

Reconfigurable Assembly System – an assembly system that is designed at the outset for a change in future application

Module – any unit, irrespective of granularity that can be combined with another unit to perform the assembly of a product

Workstation - a module that assembles one part in one orientation

Cell – a combination of one or many workstations and units based around the machines that perform the operations/ functions

Assembly Operation – the principle activity involved when two or more parts are joined together

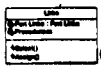
Assembly Action – a single movement or the transfer of force within an assembly operation

Process Element – a skill resulting from the combination of assembly actions

Requirements Specification – the process of extracting user needs and converting these into system characteristics

Unified Modelling Language (UML) – a communication standard for visual modelling that is now owned by IBM

Rational Unified Process (RUP) – the method recommended by IBM for transferring visual models in UML into software code.

 **Class** – a set of software objects that share a common structure and common behaviour

 **Boundary Class** – a boundary class represents an interface between the system and some entity outside the system

 **Control Class** – a class used to model control behaviour specific to one or a few use cases

 **Entity Class** – a class used to model information and associated behaviour that must be stored

 **Use Case** – a specific way of using the system from a user's perspective

Collaboration Diagram – a diagram in the UML notation that provides a view of the interactions or structural relationships that occur between classes.

Task Knowledge – prescriptive knowledge that describes which inferences are to be used to achieve a goal

Inference Knowledge – dynamic knowledge that is used to search and use domain knowledge for the execution of a task

Domain Knowledge – static knowledge that is search, retrieved and stored in a database.

ACKNOWLEDGEMENTS

The author would like to acknowledge the contribution to the work given by the following:

- Dr Svetan Ratchev for his guidance throughout the research period;
- My Parents, Jadavji Ramji Hirani and Hirbai Jadavji Hirani for their continuous encouragement and support;
- His Holiness Acharya Maharajshri Tejendraprasadji Maharaj and His Holiness Acharya Maharajshri Koshalendraprasadji Maharaj for their spiritual guidance and blessings;
- Niels Lohse, George Valtchanov and the late Anthony Chrisp for their fruitful discussions and practical help;
- Mark Jones, Gavin Murray, Geoff Turner and the team at TQC Ltd for their help in industrial use case development;
- Jonathan Armishaw at Southco. Ltd and Steven Yardley at Rexroth (Bosche Group) for their useful insight into the industrial problem;
- The Assembly-Net consortium for providing a platform for discussion at various international meetings;
- Engineering and Physical Sciences Research Council (EPSRC) for funding the research.

1 INTRODUCTION

1.1 INTRODUCTION

This chapter declares the initial reasons for the study and states the research aims and objectives. The research framework is discussed and the structure of the thesis is outlined. A summary of the research is presented.

1.2 BACKGROUND AND MOTIVATION

Automated assembly has become a key technology in areas such as electronics and mechatronics. One of the key features of assembly systems in this context is the need for reconfiguration and re-use of the assembly machines and modules to support a wide range of assembly technologies and products over sufficiently long periods of time.

From the manufacturers' point of view the need for reconfigurable customisable assembly machines and cells is defined by the requirements for increased product customisation and improved competitiveness in terms of lower cost, shorter delivery times and improved quality. A number of approaches have been reported for development of flexible assembly systems addressing the issues of computer integrated robot assembly system design (Rampersad, 1993), (Delchambre, 1996). A key factor for recent developments in assembly automation has been the need for portability, rapid specification and delivery of customisable assembly cells on demand where cells can be specified and configured over the web and delivered by different distributed module vendors (Hollis and Quiad, 1995).

This notion has formed the basis of a new European Union funded project led by Philips CFT (Evolvable Ultra Precision Assembly Systems –EUPASS), which aims to create depots of assembly modules throughout Europe. These can then be integrated and deployed for specific assembly applications as and when they are needed.

The author has observed that when developing assembly cells with highly complex modular structures designers need to translate user needs into a set of design rules and potential cell configurations. During this phase, the main architectural and behavioural requirements for a new assembly cell are collected from the user and then documented and validated. Requirements are analysed by the supplier and transformed into clear product requirements that specify assembly processes and system type and provide the link to potential cell designs (existing or new). Success in matching user requirements to potential products is dependent on how well functional and non-functional customer requirements are understood and translated into cell features.

Mostly this is done implicitly by the systems integrators, who have to cope with inconsistent information with requirements that are either incorrect, missing, ambiguous, or not specified to the correct level of detail.

It is the role of requirements engineering to help eliminate the unnecessary errors where requirements engineering is defined as *'the elicitation and formulation of requirements to produce a specification'* (Easterbrook, 1991). Requirements engineering is a dynamic knowledge intensive process involving collaborative elicitation, formulisation, analysis and negotiation of requirements (Dignum, 1999).

It also involves several stakeholders with different and possibly conflicting interests. The problems in trying to establish an adequate and stable set of requirements are outlined by Kotonya and Sommerville (1998).

Hence, there is a need for generic methods to support the interactions between different stakeholders at the early product and assembly system design stages. An important step in this process is the identification, structuring and formalisation of the requirements engineering and system design knowledge to allow different levels of knowledge abstraction and exchange.

1.3 Research Aims and Objectives

In response to the above, the research aims to provide systems integrators with the knowledge, models and tools for requirements specification of Reconfigurable Assembly Systems. This involves the achievement of the following objectives:

- A requirements specification model and methodology
- A knowledge model to support the requirements specification methodology
- A process capability model that captures the ability of different assembly operations to be reconfigured

A requirements specification methodology with supporting tools for converting user requirements into system requirements for conceptual design of Reconfigurable Assembly Systems is needed. It is a commonly accepted fact that correct specification of the problem leads to savings in time and cost at a later stage due to the prevention of expensive rework (Kotonya and Sommerville, 1998).

Formal representation of assembly knowledge has been a long term deficiency in the assembly field as operators who have the knowledge are scarcely consulted (Onori et al, 2002). An assembly process capability model will be created to help systems integrators specify system requirements accurately based on assembly characteristics and their applicability to different scenarios.

The provision of a knowledge based framework that includes a template for system users to specify their needs when ordering Reconfigurable Assembly Systems is proposed. This template takes the form of a computer program that can be easily used and kept up to date through knowledge and data support activities. Once knowledge has been stored and formalised, the organisation can still use it despite staff turnover (Scarborough, 2000).

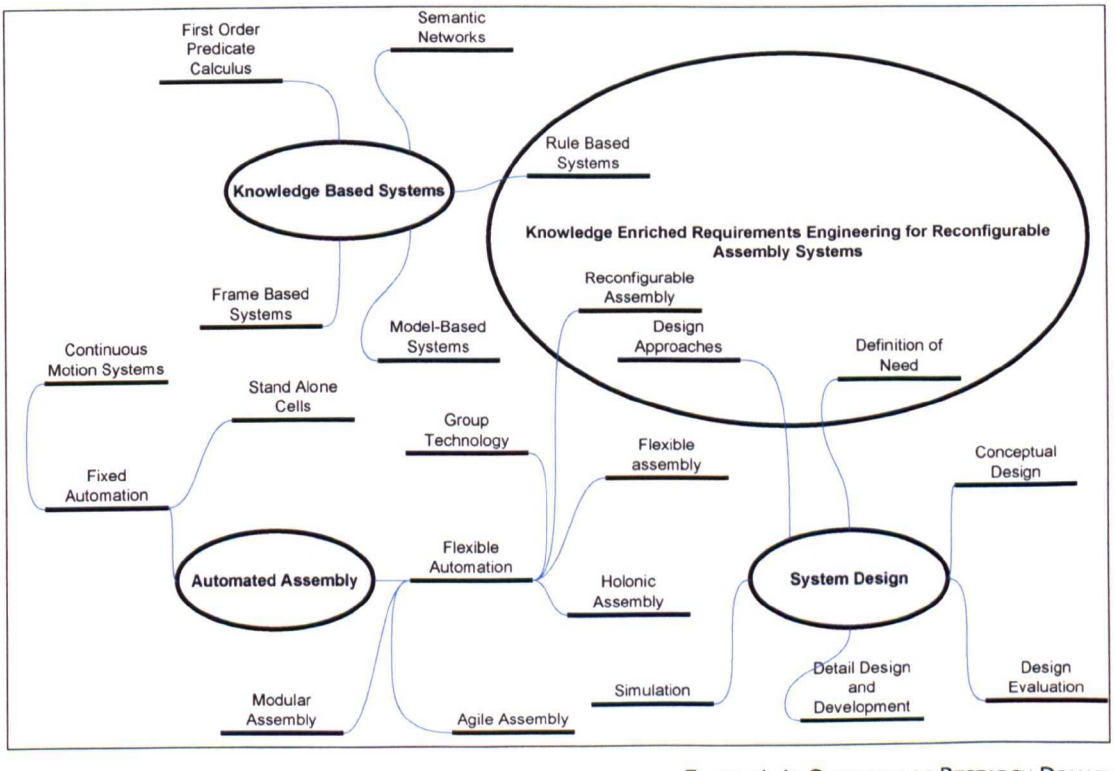


FIGURE 1-1: OVERVIEW OF RESEARCH DOMAIN

Through these aims and objectives, the research proposes to integrate knowledge in the fields of Assembly System Design, Knowledge-Based Systems and Assembly Automation to form new knowledge as per Figure 1-1.

The research draws upon system design, automated assembly and knowledge based systems as sources for previous works related to aims and objectives. A rule based system will integrate the definition of need with respect to a design approach for a one of a kind Reconfigurable Assembly Systems. Knowledge from requirements engineering and assembly system design is applied to develop new knowledge guidelines on how requirements can be specified to take into account the future implications for system reconfiguration. Moreover a methodology and supporting tools are proposed to facilitate the definition of needs and works towards finding a new design approach in the system design area. A rule based approach applicable to Reconfigurable Assembly Systems is used to do this.

1.4 Structure of the Thesis

The aims and objectives declared above have been achieved through the work carried out as part of the research. Evidence for this is presented in the remaining chapters of this thesis.

A review of related literature in requirements engineering, assembly system design and knowledge based systems is presented in section 2. This forms a theoretical background for the work including a description of related previous works found by the author in this area. Gaps in current knowledge are identified and are subsequently used to justify the present work.

The scientific method explaining how the research aims and objectives were met and the rationale behind the decisions taken for this work are presented in section 3. This includes a description of the work done and the methods used. Decisions such as the choice of software and computer language are also explained in this section.

The Assembly System Requirements Specification Model and Methodology for Requirements Specification of one of a kind Reconfigurable Assembly Systems is described in section 4. Use cases and classes are explained here at the task level of knowledge. Subsequent use of this is made in section 5.

Section 5 is Knowledge Support for Requirements Engineering of Reconfigurable Assembly Systems. Domain and inference level knowledge is described in this section, including detailed tables from the domain knowledge schema and examples of specific instances of inference knowledge.

The Assembly System Capability Model for Requirements Engineering of Reconfigurable Assembly Systems is presented in section 6. Assembly processes are described in terms of operations, actions and process elements. This forms the basis for system reconfiguration, depending on the level of modularity that is required.

A pilot environment has been created as a proof of concept. This is described with the aid of an industrial case in section 7. The general architecture of the environment is illustrated and requirements specification for a Reconfigurable Assembly System to assemble a car glove box latch is performed. System reconfiguration options are also specified with the prospect of another variant being introduced on the assembly system at a later date. A critical evaluation of the work is presented here.

A summary of the research contribution and ideas for further work are stated in section 8. This includes avenues for exploitation of the research outcomes as further work.

1.5 SUMMARY

Shortening product life cycles have put a strenuous demand on assembly systems to change along with the products that they assemble. Reconfigurable Assembly Systems are needed to achieve faster and cheaper time to market and time to volume for key products and accurate requirements specification is needed to achieve this as systems integrators (the people who supply assembly systems) need clearer guidelines on what systems users (their customers) want.

The aim of the research is to provide systems integrators with the knowledge, models and tools for requirements specification of reconfigurable assembly systems. This encompasses the creation of a Requirements Engineering Model and Methodology; an Assembly System Capability Model; a Knowledge Model; and a Pilot Environment that demonstrates the research outcomes.

2 LITERATURE REVIEW

2.1 INTRODUCTION

Advances in manufacturing technology have always been supported by tools that enable the leap to the new paradigm (see Figure 2-1). As we are now moving further into the 21st century, Reconfigurable Manufacturing Systems have been identified as one of the grand challenges for the year 2020 (Bollinger, 1998) and reconfigurable assembly is very much part of this initiative. As with the previous paradigms in Figure 2-1, reconfigurable manufacturing needs to be supported by new technologies. It is proposed that knowledge engineering and requirements engineering fit this purpose.

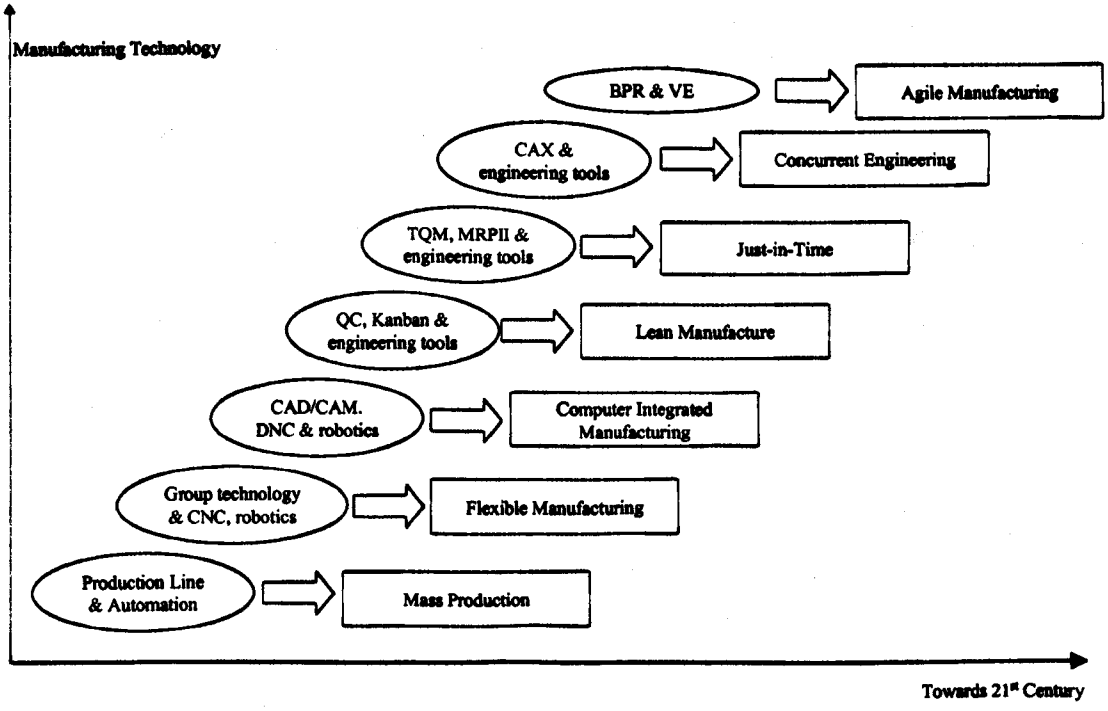


FIGURE 2-1: DEVELOPMENT IN MANUFACTURING TECHNOLOGY (*SOURCE: CHENG ET AL, 2000*)

The main aspect of this research concentrates on Reconfigurable Assembly Systems. Hence this area is explored including the theoretical background and the state of the art. Research in requirements engineering is presented to highlight the relevance of this study and the field of knowledge engineering is included to provide a sound theoretical background to the study. Knowledge gaps in the literature are highlighted and these provide a theoretical basis for the remainder of the thesis.

2.2 Reconfigurable Assembly Systems

Assembly is the process of joining together various parts to create an end product (Lotter, 1989). For this to occur there are various trade-offs to be considered and these are summarised in Figure 2-2.

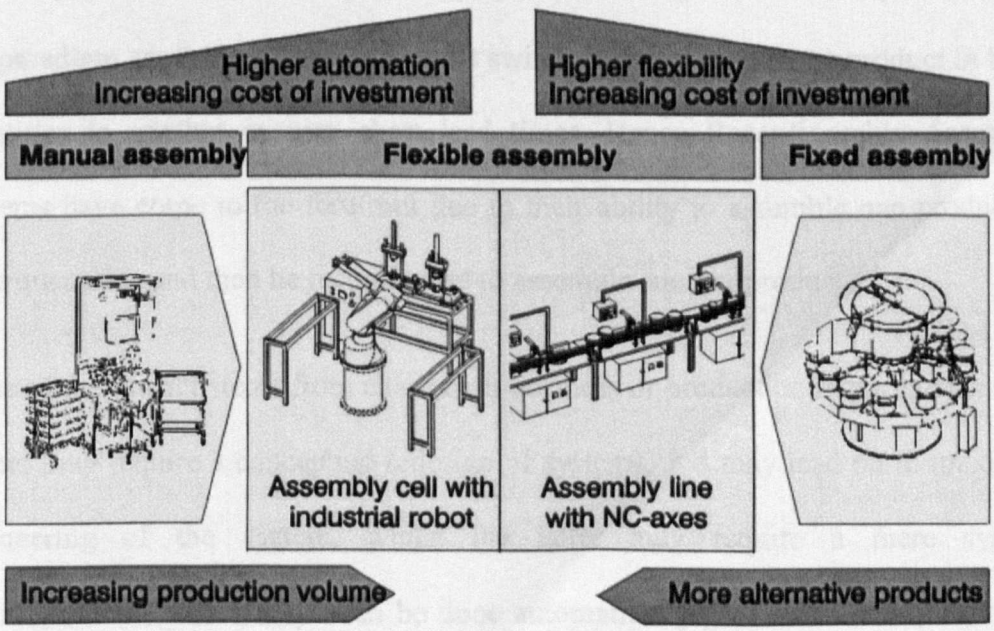


Figure 2-2: Tradeoffs in Traditional Assembly (*Source: Feldmann & Rottbauer, 2000*)

Assembly Systems are classified as manual, flexible or fixed systems. Manual assembly can deliver a wide range of products with a relatively low investment, but this can only be sustained for low volume production. Fixed assembly caters for high

volume production, but requires a large amount of capital investment on automated assembly equipment (Feldmann and Rottbauer, 2000).

However, flexible assembly is seen as a suitable trade-off between the two extremes by providing volume production at reasonable prices. Johansson (2002) reviews the developments in flexible assembly, including the contribution of specific systems to the flexible automated assembly concept. These are analysed in terms of static and dynamic flexibility where static flexibility refers to the ability of the system to deal with different variants and static flexibility refers to the ability to change between different products and capacity constraints.

Johansson states that there is a trend towards modularisation and more dynamic flexibility. Hence the flexible assembly system is being superseded by variants of this paradigm as production requirements switch from producing one product in large quantities to another in very short lead times. Hence Reconfigurable Assembly Systems have come to the forefront due to their ability to assemble one product in large quantities and then be reconfigured to assemble another product.

Some of the impact stems from changes in business or production requirements. The former may require a conceptual redesign of systems, and may lead on to major re-engineering of the system, whilst the latter may require a mere system reconfiguration, which may even be done automatically provided the requirements have not changed drastically. Furthermore, the latter is easier to predict and accommodate through traditional cell design methods, but the work by Monfared and Weston (1997) presents the case for producing a cell design structure that can be flexible to both types of changes.

Their work also identifies three areas where system design and construction fails. These are:

- Inflexible links between resource elements.
- Specialist systems, which cannot support changing requirements.
- Systems of limited practice parochial scope.

The article develops a meta-model with the aim of retaining the principle of supporting flexible mapping between logical models of a cell and its physical elements. The challenges highlighted above are overcome through facilitating:

- Re-engineering of cells systems in response to changing business needs.
- Reconfiguration of cell systems in response to changing production needs.
- Reuse of cell components.

Hence system reconfiguration is a viable solution to flexibility requirements problems. This is explained vividly by Chick et.al (2000) who state that the reconfigurable manufacturing system is created from a set of basic process modules – hardware and software – that can be rearranged quickly and reliably. The following five characteristics are essential for this (Koren and Ulsoy, 1997):

- Modularity
- Convertibility
- Customisation
- Integratibility
- Diagnosability

Furthermore, a system that exhibits these characteristics will allow dramatic reduction in launch time of both new systems and rebuilt systems and achieve system upgrading relatively quickly and inexpensively by upgrading one or two modules at a time rather than replacing the entire system.

An example of a reconfigurable machine is as follows: suppose that a machine is capable of milling and drilling but not turning. A reconfiguration option could be the ability to quickly and easily introduce a turning capability when needed in the future. Some measurement criteria are also suggested here to aid in the quantification of different design ratings and hence the ranking of different designs based on their feasibility.

Two features of reconfigurable manufacturing systems in considering future changes are also presented:

- System reconfigurability: consideration of the ability to rapidly reconfigure the manufacturing system, to alter the product flow in order to affect throughput, quality and other attributes.
- Equipment reconfigurability: the use of contracts to permit functionality and/or capacity to be added, and paid for, at a time or times in the future determined by the buyer.

The ideology of Reconfigurable Assembly Systems is encapsulated by Huff and Edwards (1999). Their work declares that a truly reconfigurable system is only possible if the system is developed around a base platform, which provides generic production resources and enablers. Moreover a Reconfigurable Manufacturing System has a three-layered approach:

- Base Layer – provides generic production resources and application development toolsets
- Process Layer – the base platform and the process building blocks provided determine the functional capability of the system
- Product Layer – specific product hardware and information which identifies processes and production sequences required to produce specific products.

Design for reconfigurability has been presented through research at the Intelligent Systems Lab, University of Iowa. Reconfigurability is defined here as *"the ability of a manufacturing system to be rearranged at a low cost and in a short time for producing a variety of components or products."* (Huang and Kusiak, 1998) This is elaborated into two categories: dynamically and statically reconfigurable systems. Dynamically reconfigurable systems are achieved through routing of operations and statically reconfigurable systems through system design. Design for reconfigurability then becomes a matter of designing manufacturing systems or components or both for reconfiguration. The aim is to minimise the movement of machines and transport mechanisms to achieve manufacturing efficiency.

This may involve changes to the system such as:

- rearrangement of machines
- reassignment of operators
- retooling of machines

The design of such systems can be categorised into six areas:

- Proper layout of manufacturing cells

- Manufacturing processes
- Design of factory floor
- Modular design of equipment and tools
- Limiting constraints imposed on machine locations
- Multidirectional flow of material-handling carriers

In addition to this geometric features, dimensions, components selected, precedence relations, tolerances and material specifications must be revised at the component level to incorporate reconfigurability.

Basic components interact with distinct modules, resulting in different product variants where a module is an independent unit with specific hardware, electrical and mechanical interfaces that allow it to perform a defined function. According to this definition an assembly module can be an assembly cell or an assembly workstation or a pneumatic or hydraulic unit. This accommodates the possibility of modular assembly to exist at different levels of abstraction (Lohse et al, 2003). Kusiak identifies five types of modularity:

- Component swapping – where two or more basic components may be swapped, creating different variants belonging to same product family
- Component sharing – where same basic components create product variants belonging to different product families
- Bus – whereby a module can be matched with any number of basic components. This allows for variation in the number and location of basic components

- Selection – based on standard components
- Fabricate-to-fit – where modules differ in a limited number of parameters.

Modularity depends on the similarity between physical and functional architecture and the minimisation of incidental interactions between physical components. This implies that two types of relationship are involved: similarity of functional interactions within a module and suitability of inclusion of components in a module.

The essence of modular systems is that the designer spends expensive design time on the unique parts of the machine, not those that have already been designed. It takes the form of a Design→Create→File→Update sequence. Modularity is hence an ideal way to support both the schools of thought when it comes to building assembly machines (Jordan, 1997):

- Build something totally unique the likes of which has never been seen before.
- Bundle together a known selection of tools to perform a specific task.

Rizzi et.al (1997) carry out a study looking into the design of a minifactory using interchangeable modules. The case for design and programming using a graphical interface is presented here due to the benefits of simulation through off-line evaluation properties. Their system brings advantages in terms of modularity; robustness; scalability; and ease-of-use.

A modular conveyor system that can be used to transport parts and components between modular stations has been designed by Ho and Ranky (1997). This system adopts dynamic re-routing; real-time changes; simultaneous assembly of different products; copes with unplanned events; minimises transport time; allows object-

oriented design; and incorporates set-up and modification of hardware and software – negating the problems with conventional conveyor systems.

The key performance measures in either case are speed and flexibility. The features that are required to achieve this are highlighted by Heilala and Voho (2001).

- Human friendly ergonomics and info tools
- Modular generic building blocks with standard interfaces
- Rapid implementation, fast deployment and re-deployment, time-to-market
- Scalable, adaptable to varying life product cycle volumes
- Reusable, redeployment for different product models and families, product life cycle, economics
- Agile, adaptable to individual customer needs, time-to-customer, mass customisation
- Reconfigurable, ability to arrange modules for different objectives
- Flexible robotic cells
- Information technology integration, Ethernet to the factory floor

The NEMI plug & play factory project (Dugunske et al, 2000) aims to incorporate these features by using standardised SMT equipment. As all equipment is then physically interchangeable due to standard interfaces and devices, the control aspects are configured through web enabled messaging using the XML message exchange format.

The Department of Precision Engineering at The University of Tokyo has developed the Holonic Assembly System based on a "plug & produce"(Arai et al, 2001) concept drawing inspiration from the efforts of NEMI and the "Plug & Play" concept from the information technology industry. The aim is that a new manufacturing device should be instantaneously reconfigured to work within a workspace upon installation with little or no down time. This is done through a system of Holonic control, where each device has its own control mechanism programmed into it (execution holons). These are triggered by management holons, which could be of three types: task; process; or operation. The system manager creates tasks, which are decomposed onto execution commands through a complex holarchy until they are in an executable form. The advantages of Holonic Manufacturing Control over traditional hierarchical systems are revealed by Bongaerts et.al (2003). The holonic system of control is more predictable and easier to coordinate between separate entities simultaneously and it safeguards the robustness of distributed control.

Furthermore, a rapidly reconfigurable robotic workcell system has been developed by Chen (2001). Here the system consists of four elements including:

- modular reconfigurable robots
- reconfigurable simulation and control software
- supplementary workcell device
- workcell control software.

Chen identifies that the major emphasis in this field of study is the high uniformity in the design of the assembly modules. This is illustrated by the Agile Assembly

Architecture (AAA) where an entire assembly line may be composed of identical units - each robot is equipped with different grippers and programmed for different actions. Each station is capable of four degrees of freedom motions. End effectors are designed such that they can be easily interchanged among the different stations as and when needed (Hollis and Quiad, 1995).

In essence the project looks at designing a minifactory using interchangeable modules. Design and programming are performed with a graphical interface. This supports the case for simulation due to the advantages of off-line evaluation. The AAA brings advantages in terms of:

- Modularity;
- Robustness;
- Scalability;
- Ease-of-use.

The approach used here results in a very similar output to the Minifactory project (Muir et al, 1997) which consists of a collection of mechanically, computationally and algorithmically distributed robotic modules referred to as agents. Each agent is an independent entity executing its own program. Task based abstractions allow agents to be programmed with a minimal level of dependence on the explicit behaviour of their peers. Each agent is able to robustly execute their task directed programs by ensuring their proper calibration with respect to relevant features in the Minifactory.

The above two combine with the ability of agents to provide accurate physical and behavioural models to the simulation system.

2.2.1 Current Trends

The American National Research Council has identified reconfigurable manufacturing systems as one of its visionary manufacturing challenges for the year 2020. This is defined as *"adaptable, integrated equipment, processes and systems that can be readily reconfigured for a wide range of customer requirements for products, features and services..."* The report goes on to say that *"Ultimately, a library would be developed of reusable processes and sub-processes for building and reconfiguring manufacturing systems."* (Bollinger, 1998)

Although the NRC is generally talking about manufacturing, there is no reason to see that these challenges are not relevant to assembly as assembly is a specific application of manufacturing.

Researchers at the University of Michigan (Mehrabi et al, 2002) carried out a survey covering flexible and reconfigurable manufacturing. Findings from the survey revealed that *"two-thirds of respondents stated that they did not believe that FMS is living up to its promise across all manufacturing."* The specific areas of dissatisfaction were ramp up time and investment in technology and functionality that was not utilised.

Reconfigurable manufacturing was highlighted as the future of flexible manufacturing as it confronts these issues although it was emphasised that *"modular machines and open architecture control systems must be developed for RMS to be*

realised." Important criteria for the modular machines were also identified in the survey. The results revealed that *"system design time, machine installation, ease of adding new features, ease of upgrading technology, part quality and accuracy, the ability to customise system features, multifunctionality and cost"* were the important criteria to be considered in the developed of RMS. However *"Software issues represented probably the single area of greatest concern for the successful development of RMS technology."*

Recent trends in this area have been explored by the Assembly-Net consortium, resulting in the publication of the European Precision Assembly Roadmap for the year 2010 (Onori et. al, 2003). A summary of the roadmap findings and its bearing on the research is presented in 2.6

2.3 ASSEMBLY SYSTEM DESIGN

The available literature in assembly system design is very broad and covers many different aspects of assembly. For this study only the relevant literature in the area of flexible and reconfigurable assembly is reviewed.

The dictionary definition (OUP, 1996) of assembly is "*the action of assembling component parts*" or "*a unit consisting of component parts*", whereas the definition of a system is "*a complex whole; a set of things working together as a mechanism or network*".

Ye and Urzi (1999) define assembly as "*a group of parts together serving one purpose*" whilst Burbridge (1989) defines assembly systems in two ways:

"*A small organisational unit which completes assemblies in a continuous flow and is provided with all the facilities it needs to do so.*" The second way consists of a series of "*Yes/No*" questions related to the state of a group of workers based on behavioural science methods.

Yet another different approach is adopted by Rampersad (1993), who defines an assembly system in terms of functions and tasks, whereby the function is to carry out an "*assembly of product parts into final composites, which are required by the environment*" and the task, is to "*execute assembly operations in order to fulfil the system function.*"

A more detailed definition is presented by Stadzisz and Henrioud (1998) who claim that: "*The assembly processes of a product consist of a set of operations, which fit components and subassemblies together by placing some faces of the components in contact.*" Hence each assembly operation is a set of four elementary tasks:

- Fixturing the primary constituent
- Feeding
- Grasping
- Positioning of the secondary constituent

The implications for assembly design are thus derived by Stadzisz and Henrioud. Roughly assembly design involves:

- The establishment of the functional requirements from the analysis of the customers needs and expectations

- The design of the products to satisfy these requirements
- The design of processes and the system to assemble the designed products.

The causal dependence here in the form of function → product → assembly process is worth noting. This is translated into a set of activities encompassing the following that can be applied at each design iteration:

- Generation of feasible assembly plans
- Discard of non-promising base components
- Discard of non-promising assembly direction
- Evaluation of the assembly difficulty
- Evaluation of the required flexibility
- Generation of design advises based on the assembly evaluation
- Generation of design constraints based on the assembly processes decisions.

The article referred to here highlights the deficiency of traditional processes where design is viewed as a hierarchical decomposition process. Thus decision trees are created where tasks are decomposed but the relationship between the tasks is ignored. In order to counteract this, an integrated approach is proposed involving the simultaneous development of a functional model, a physical model and a model of the assembly process.

Broman and Eskilander (2000) highlight an alternative method in their four steps for an automatic assembly system design:

- Comprehensive questions – covers overall input parameters such as output quality, product variants and physical attributes.

- Mapping the product – product structure is surveyed to determine the required assembly operations. The DFA2 technique is employed to clarify the assembly sequence and highlight necessary operations.

- Choice of technical solution – derived from product mapping to existing knowledge.

- Layout of the system – compilation of technical solutions to form a system.

The same article also establishes that the successful design of a technical system relies upon:

- proper examination of problem to solve
- creative, experienced knowledgeable environment
- ability to assess design proposed performance
- ability to interpret the results of such and analysis

They proceed to state that when designing an assembly system, it is important to establish the correct starting point for the process as alternative concepts are created as a consequence of three aspects:

- Manufacturing strategy
- Experience from earlier assembly systems
- System requirements.

An alternative approach is presented by Nkasu and Leung (1995), who have divided the work activities involved in the design of a manufacturing assembly system into several logical practical physical components including: process design; assembly line balancing; test strategy; yield management; material handling; maintenance policy; WIP management; parts procurement; parts feeding; human resources; assembly system size; and information system design.

Consequently a Computer-Integrated Manufacturing System Design (CIMASD) is derived where a list (A) is constructed that tabulates the total number of tasks which immediately precede each given task already on the system. All the tasks without any preceding tasks are placed onto another list (B -available list). Then tasks with times less than the available station times are put into list C (fit list).

Heuristic rules and strategies have been derived by Ye and Urzi (1999) for the purpose of obstruction prevention, ease of handling, efficient operation and product safety. At this point it is important to consider the shortfall of heuristics in that they rely on the knowledge of the engineer to supply information upon which the heuristics may be applied.

A computer is a much more powerful tool for evaluating different design ideas and the role that computer simulation has to play in the design process has been explored in the literature. For instance Chan and Jian (1999) reveal that computer simulation can be used to:

- Determine equipment needed to achieve planned capacity
- Identify potential problems, such as bottlenecks

- Investigate alternative plant layouts
- Determine required equipment performance
- Determine impact of equipment reliability
- Test various hypotheses

Henceforth, designs can be evaluated with respect to:

- Ability to meet production requirements
- Labour and machine utilisation
- Cells capacity to meet increased demands

Tichem et.al. (1999) have identified that there is a need for fundamental steps forward in the design of products, assembly processes and assembly equipment. These three aspects have to be considered concurrently, not sequentially. However, external pressures mean that a system must outlive the product/product range for which it was originally built, which requires further inroads in the research domain - reconfigurability.

The most referred to method for designing assembly systems found by the author in the literature is Rampersad's (1993) integrated and simultaneous design for assembly systems. An outline of the model is presented in Figure 2-3. The advantage of this model is that robotic assembly systems are designed whilst simultaneously looking at the product, the assembly processes and the assembly system.

An alternative model is presented by Ranky (1998) who advocates the emergence of distributed systems and the challenges and advantages in terms of communication

and generation of ideas. The recommended system gives the designers, programmers and users the freedom to learn what they want, when they want and how they want for themselves as well as the flexibility to make modifications and gain valuable feedback regarding solutions to design problems.

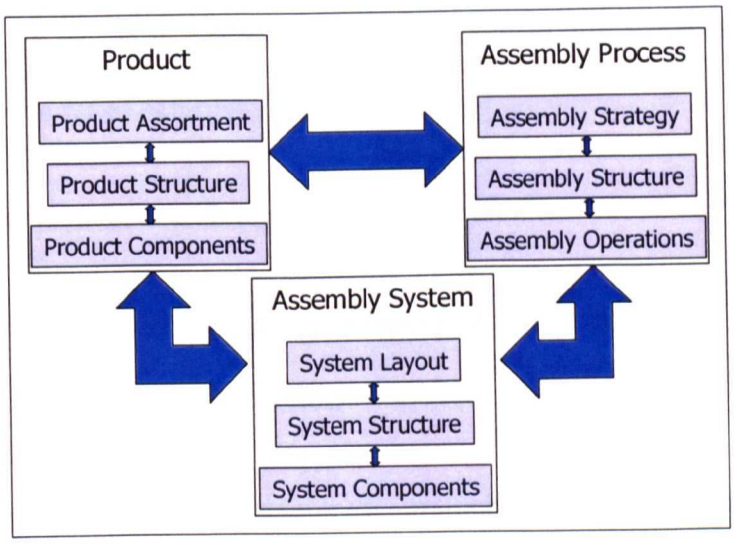


FIGURE 2-3: INTEGRATED DESIGN MODEL (*SOURCE: ADAPTED FROM RAMPERSAD, 1993*)

Manufacturing research in this field can be explained under two domains (Chau et al, 1995):

- **Assembly modelling** includes the representation and modelling of individual parts; positional relationships between parts and; mating conditions
- **Assembly planning** deals with the creation of an assembly plan and evaluates the mechanical, electrical, control as well as cost feasibility.

Numerous methods of assembly planning have been presented in the literature and most of them follow a hierarchical structure as outlined by Chakraborty and Wolter (1994). The main advantage of such a system is that it allows easy tracking of changes to facilitate reuse.

From the Lucas perspective the ideal manufacturing system is designed around two core processes: manufacturing operations and product introduction processes with support processes. Each process has the following ideal attributes (Mason-Jones et al, 1998):

- Converts a set of inputs in an integrated way to add value and produce an identifiable set of outputs;
- Has clear interfaces with other processes;
- Is controlled by a natural group or team in a seamlessly linked and integrated group of skills and competencies;
- Starting from a set of customer requirements each process delivers a high quality total product or service to a competitive target cost and lead-time;
- Each process can be described and identified by a diagrammatic analysis technique that highlights the value adding operations that make up the process.

Flexibility and delivery have been identified by Kumar et.al (2000) as the most dominant priorities for both global and domestic manufacturing firms. This has resulted in many organisations reviewing, redesigning, and reconfiguring their manufacturing systems to accommodate the significant flexibility and fast-delivery capabilities into their operations.

However, the fundamental development that distinguishes flexible assembly from reconfigurable assembly is that of "modularisation" where each assembly module

has a built-in functionality that can be coupled with other modules to provide an overall flexibility to the integrated system. The advantage of such a system lies in the notion that each module is a self-contained unit, and hence a change in one module has minimal effect on other modules within the system.

This is explained in more detail and applied to product family architectures by Jiao and Tseng (2000) who claim that "*a product's architecture is often thought of in terms of its modules. A module is a physical or conceptual grouping of components. Modularity is the concept of decomposing a system into independent parts or modules that can be treated as logical units. Modularity has been defined as the relationship between a product's functional and physical structures such that:*

- *there is a one-to-one or many-to-one correspondence between the functional and physical structures; and*
- *unintended interactions between modules are minimised.*"

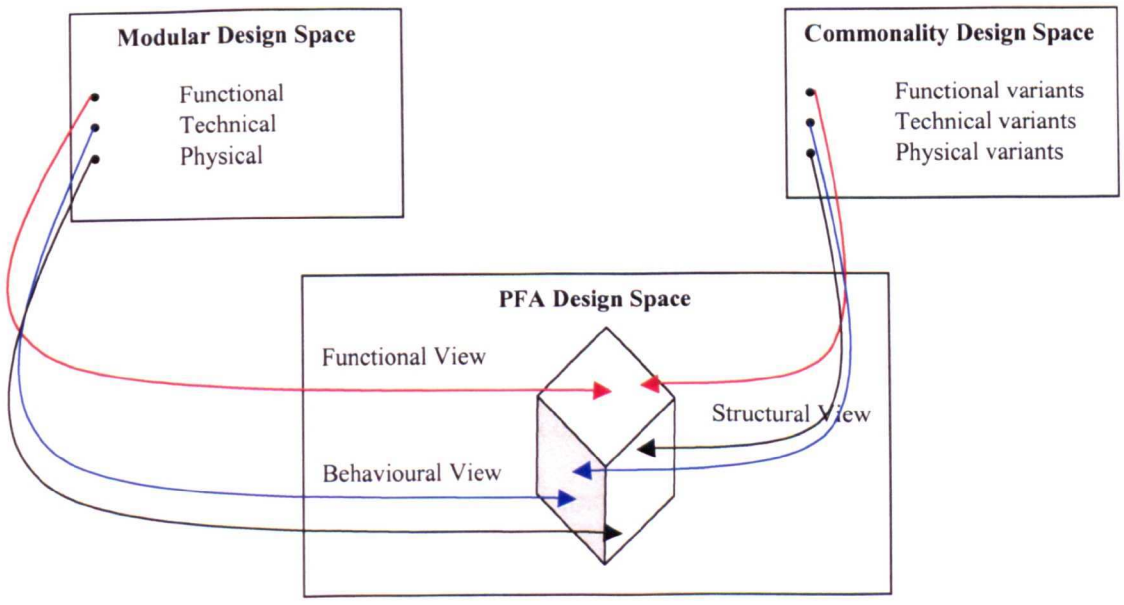


FIGURE 2-4: THE PFA DESIGN SPACE

A PFA design space (Figure 2-4) has henceforth been developed to tackle the problem from three perspectives:

- Functional view – embodies product line structure in terms of different customer groups; functional features and their relative importance/priority for every customer group; and classification of functional feature instances for customers within each customer group

- Behavioural view – contains modules and modular structures defined in terms of technical parameters corresponding to specific functional features instead of physical components and assemblies. The purpose is to highlight differentiation (variety) in product design resulting from different solution technologies applied to meet diverse customer needs. The variation resulting from manufacturing concerns is dealt with by the structural view of the PFA. Issues regarding the technical modelling of a technological solution include documenting technical parameters and the mappings from functional features to technical parameters; determining technical modules by minimising design couplings; and establishing modular structures for design synthesis.

- Structural view – represents product information by a description of the physical realisation of a product design and is strongly related to the construction of the product. This consists of various types of components and assemblies in order to realise technological solutions/product technologies generated in the behavioural view.

Flexible assembly system design is based around the design of two basic mechanical systems: the manipulator and the material handling system (Edmondson and Redford, 2002). Nakase et al (2002) approach the problem from a management perspective whereby decision making is based upon calculation of feasibility in two cases: the design of completely new facilities and the modification of existing facilities. In both cases a 2-stage design process is followed which encompasses consideration of:

- Economic Traffic
- Economic Buffer and Lead Time (Reliability)

These are considered from two perspectives: Market first and Production first and are then fed into a production matrix for further evaluation. Although the approach is useful in terms of determining the number of assembly stations required and cycle times at each station, the technical detail regarding achievement of the required cycle times is not considered.

The closest work to structuring assembly processes has been performed by Vos (2000), who has analysed assembly processes and compiled a list of ten basic operations from which all assembly processes are built. These operations are combined and parameterised to develop flexible assembly systems from a set of Flexible Assembly System elements, which are combinations of Products, Assemblies, Parts, Interchangeable Tools and Non-interchangeable Manipulators.

Furthermore a design strategy for Reconfigurable Manufacturing Systems has been suggested by Abdi and Labib (2003). This categorises products to be manufactured

in the system into a set of product families and assigns these products to manufacturing system elements. Manufacturing system planning occurs at three levels of decision making based on long, medium and short terms. These are evaluated according to objectives in terms of responsiveness, product cost, product quality, inventory and operators' skills. Although this framework is coherent for a known set of products and product families it is much more difficult to build reconfigurability into a system at the outset when all the products are not known.

Analysis of the literature referred to above revealed that all the works looked at assembly system design from a product perspective and how the product could be assembled. Little or no attention was paid to the business case for assembling the product and there were no articles that actually addressed the flexibility issues, where examination of the problem from a process perspective is vital.

2.4 REQUIREMENTS ENGINEERING

2.4.1 Requirements in the Assembly System Design Process

Traditionally, requirements engineering is defined as 'the elicitation and formulation of requirements to produce a specification' (Easterbrook, 1991). Hence, requirements engineering refers to activities of gathering and organising customer requirements and system specifications, making explicit representations of them, and making sure that they are valid and accounted for during the course of the design lifecycle of the product. The requirements engineering process involves a clear understanding of the requirements of the intended system. This includes the services required of the system, the system users, its environment and associated constraints. This process

involves the capture, analysis and resolution of many ideas, perspectives and relationships at varying levels of detail. (Ratchev and Hirani, 2001a)

Current research in this field has been summarised by Darlington and Culley (2002).

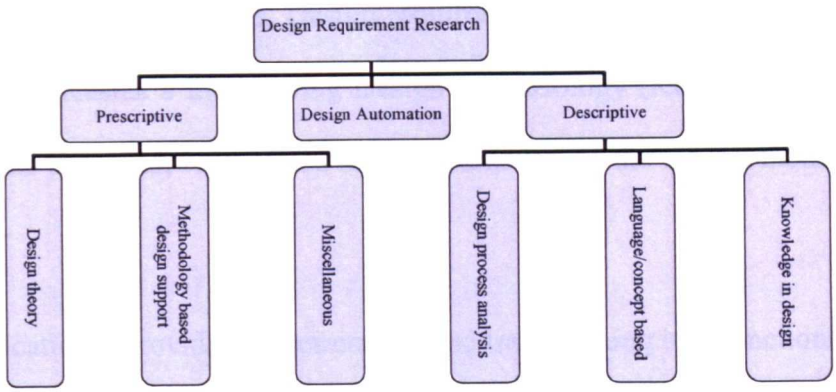


FIGURE 2-5: TAXONOMY OF ENGINEERING DESIGN RESEARCH (*SOURCE: ADAPTED FROM DARLINGTON & CULLEY 2002*)

Although research in these fields has been performed for general design projects, these cover specific areas for each specific case. No study has been reported that encapsulates all the areas with application to a single field, i.e., using prescriptive methods to provide automated or semi-automated design together with descriptive means.

Furthermore the differences between the application of requirements engineering in software and engineering disciplines are outlined where engineering design aims to "*arrive at a complete, concise and correct description of the design need, expressed in natural language.*" (Darlington and Culley, 2002) This involves the formalisation of design requirements into a structured methodology whereas software requirements are more flexible and do not need to be processed in this way.

Bray (2002) highlights the need for Requirements Engineering and provides a basic definition of the principles. Furthermore, the importance of design requirements is acknowledged by Stadzisz and Henrioud (1998) who describe how requirements are filtered through the design cycle as products and product families are created to serve the wants and needs of customers. Design requirements are captured and processed in three stages in Kusiak's Engineering Design methodology (Kusiak, 1999) where requirements are specified, represented and synthesised into the overall model of the design project:

- Specification – providing requirements and transforming into functions
- Representation – where components are assigned to functions
- Synthesis – of the above into overall model of design project

Requirements are decomposed into sub-requirements and the number of levels of sub-requirements depends on task complexity and the detail required. Functions, as well as requirements are specified for each domain. Each requirement may be satisfied by more than one functional module and vice versa – i.e., many-many relationship. Kusiak then moves onto satisfying the individual requirements following the QFD method. This approach is very much geared towards modular products for which the requirements seldom change and not systems that need to be reconfigured many times throughout their life.

2.4.2 The Requirements Engineering Process

In his article on requirements engineering and management, Carr (2000) defines requirements engineering and management as *"the process of discovering, documenting and managing system requirements."* This involves eliciting understanding, describing, validating and managing system requirements.

The role of requirements engineering is to provide an abstract solution for a design problem. Moreover *"a good set of requirements defines precisely what is wanted, but simultaneously leaves the maximum space for creative design."* (Stevens and Martin, 1995). These requirements have to reflect the customer's expectations of the system.

The meaning of requirements is examined by Jackson (1997), who claims that system failings are often the consequences of inappropriate requirements and that the complexities and subtleties in requirements specification must be addressed to avoid system failure. The interaction between the machine and its environment is the key aspect to consider here and all system properties must be defined using these terms.

One method of having concrete requirements to work from is to develop formal models, tools and techniques (Jackson et al 1995). This has been implemented in software writing where object oriented specifications are drawn up and held within a storage facility. Links between the various objects are defined as part of the requirements specification. These are subsequently searched by a query language.

A database system has been designed by Jiao and Tseng (1999) which facilitates easy storage and retrieval of requirements data. Product specifications can be specified,

stored and retrieved through this database shell, but no traceability is inherent within the system.

An approach for the specification of assembly systems has been proposed by Chau et al (1995). However, this study adopts a static perspective in that it does not include the scope for future modifications to the product being assembled. The work concentrates mainly on generating an assembly tree and does not proceed to specification of requirements for assembly equipment.

The distinction between user requirements and system requirements is that user requirements define what the user wants to do with the system whereas system requirements explicitly define the properties that must be part of the system to satisfy the user requirements.

User requirements are owned by the end users of the system. These encompass:

- Product perspective – a description of the system and its context
- General capabilities – which capabilities are required and why they are needed
- General constraints – which constraints are applicable and why they exist
- User characteristics – who will use the product and when
- Operational environment – what conditions will be like where the system will be applied
- Assumptions and dependencies – the assumptions on which the requirement depends

In contrast to this system requirements include (Stevens et al, 1998):

- Descriptive elements – including terminology and assumptions
- Functional or behavioural breakdown – how the system will achieve the user requirements
- Performance – attributes on functions provided by the system
- Internal interfaces
- Non-functional requirements – generally terms of contract
- Interface control documents – one for each external system, including support and production systems
- Traceability to user requirements

Parallels can be drawn with the manufacturing subsystem design method developed by Martensson, P (2000). This work reports on the relationship between functional requirements, design parameters and the process domain where design parameters are derived from the functional requirements of the system. These are then used to select processes from the process domain. The functional requirements define what the user wants the system to do and the design parameters provide a formalised statement of how that can be achieved. For example if the user wants a fast car, then this can be formalised in terms of chassis shape, engine horsepower, etc.

The NIBA project (reported in Fleidl et al, 2000) recognises the importance of requirements and the need for better analysis of user requirements. As user requirements are usually elicited in natural language, a linguistically based method

for analysing these requirements is desirable. This approach decomposes the syntax of requirements statements and analyses sentences to ensure that they are syntactically correct. Although this provides a useful basis for completeness and consistency checking, no further work has been reported that uses the output from this.

The issue of completeness and consistency checking of requirements has been tackled by Sinha and Popken (1996). They employ an agent based approach to decompose user requirements. System requirements are specified by engineers and these are analysed for consistency and completeness with respect to a list of predefined attributes.

However, Macaulay (1999) identifies that requirements engineering is largely a human intensive task and is therefore influenced by the role of the facilitator as per her seven layer model, which charts political, social, personal, method, activities, technology and environment as the influential factors. From this it is evident that the role of the human in requirements engineering has to concentrate on those aspects that are suited to humans and that tools need to be created that empower humans to participate in requirements engineering activities.

Requirements reuse has been explored by Lam (1997). In this avionics case, two levels of reuse are possible for full authority digital engine controllers: common requirements for a functional area and; common requirements for a functional area specific to an engine mark but not a variant. Hence a domain analysis was conducted consisting of:

- Understanding the domain by reading background material, existing system documentation, and by speaking to domain experts and;
- Identifying frequently reoccurring requirements in the domain and use abstraction to develop truly generic and reusable requirements.

The result of this activity was:

- List of issues, which appeared in the starting domain in terms of requirements focal points.
- Set of generic requirements, which can be reused in the requirements engineering processes for new starting systems.
- Choice sets for particular generic requirements representing standard configurations of the requirement.
- Factor-in and factor-out lists, which describe abstraction information useful for future analysis in the domain area.
- Personal gains in knowledge about how starting systems work and how their requirements are specified at RoSEC.

2.5 KNOWLEDGE ENGINEERING

A knowledge intensive firm refers to a company where most work can be said to be of an intellectual nature and where well qualified employees form the major part of the workforce. (Robertson and O'Malley, 2000)

Knowledge is perhaps best understood as multifaceted and multi-layered, comprising cognition, action and resources as well as social networks and knowledge management is considered a general management panacea for managers seeking competitive advantages in the global marketplace. However, it is increasingly being viewed as a product of the IS/IT industry as this accounts for approximately 70% of all themes. Scarborough (2000) identifies that there are many articles in the literature focused on developing and implementing knowledge management databases but the most dramatic improvements in the knowledge management capability of an organisation are human and managerial. Some believe that knowledge management is about stockpiling workers' knowledge and then making it accessible to others via searchable applications, hence the use of computers.

In the strategic context, organisations must adjust their capabilities to a constantly changing complex external environment (Martenson, M, 2000). To create a knowledge management strategy, an organisation needs to build systems for capturing and transferring internal knowledge and best practices. Knowledge management can be seen as a way to:

- improve performance, productivity and competitiveness;
- improve effective acquisition, sharing and usage of information within organisations;
- a tool for improved decision making;
- capture best practices, to reduce research costs and delays;
- become a more innovative organisation.

However this study found that culture played the pivotal role in the success or failure of a knowledge management initiative.

2.5.1 Knowledge and Knowledge-Based Systems in Industry

"*In a 1989 survey, several Fortune 50 CEOs agreed that knowledge is a fundamental factor behind an enterprise's success and all its activities.*"(Wiig, 1997) In this article, Wiig categorises knowledge management into four areas: Governance Functions; Staff Functions; Operational Functions; and Leveraging Functions. Different foci and goals for types of application are also stated, however the focus taken in this review is concerned with Leveraging Functions and how knowledge can be used to improve requirements engineering.

The benefits of knowledge management are echoed by McCampbell et al (1999), who state that knowledge management has played a key role in transforming the fortunes of at least one consulting firm in the US. This has been performed through the sharing of resources such as documents and presentations stored within a centralised knowledge base. A critical review of knowledge management has been written by Martensson, M (2000). In this paper Martensson states that the success of knowledge management depends on the ability of an organisation to turn the intangible knowledge of employees into tangible knowledge that can be shared throughout the organisation. Furthermore a wide spectrum of viewpoints on knowledge management methods has been examined by McAdam and McCreedy (1999). This study concludes that knowledge management initiatives can be either mechanistic - forming an intellectual capital approach - or social – assuming a social constructionist approach. The practical implication of this is that knowledge is either

stored centrally through database structures or gathered by consulting other people who have been involved in similar situations.

A more recent review by Liao (2003) examines 243 articles in knowledge management and classifies these into seven categories:

- Knowledge Management Frameworks
- Knowledge-Based Systems
- Data Mining
- Information and Communication Technology
- Artificial Intelligence/Expert Systems
- Database Technology
- Knowledge Modelling

Of these expert systems are of particular interest to this research. Expert systems aim to capture human knowledge and store this knowledge as a set of rules. Rules are fired for relevant applications according to the goals that need to be achieved.

A definition of expert systems is supplied by Metaxiotis (2003) who declares that "*an [expert system] is a computer system containing a well-organised body of knowledge which emulates expert problem solving skills in a bounded domain.*" This comprises a knowledge base, inference engine and a user interface. Knowledge is stored in the form of facts, data and heuristics, the inference engine performs knowledge functions within the system and the user interface allows the user to manipulate the knowledge through the inferences. The work reveals applications of

expert systems in areas such as banking and marketing, but their work fails to identify any expert systems in the engineering domain, although there was research performed in this area in the early nineties (Valckenaers, 1993 and Archibald & Petriu, 1993)

One of the reasons for the reluctance to exploit expert systems for engineering applications is the limitations in the codifiability of knowledge (Cowan, 2001). This is because "*writing an expert system is an explicit attempt to transfer knowledge from a human to a machine*" It is difficult to do this as experts have difficulty in defining the knowledge that they use to make decisions. Moreover codifying knowledge involves three distinct aspects: creating models; creating languages; and creating messages.

The greatest bottleneck in this process is that of knowledge acquisition. One way of overcoming this is to automate the knowledge acquisition process. The method suggested by Huang et al (2001) involves deriving fuzzy rules from neural networks. Each data item is given a fuzzy weighting (small/medium/large) according to several criteria. This is then processed through rule trees in a neural network to find the meaning of the data. The approach is quite powerful, but only applicable to a set number of cases as defined in the neural network.

Another method for knowledge acquisition is suggested by Chan et al (2003). It is stated that knowledge acquisition has three stages:

- Knowledge elicitation - obtaining knowledge from an expert
- Knowledge analysis – understanding the expert knowledge collected

- Knowledge representation – formalising the knowledge gathered so that it is in a usable form.

This is implemented in Chan's Inferential Modelling Technique (IMT), which acquires knowledge about the process largely based on the KADS methodology (Schreiber, 1993). The difference between the two methods is that the IMT uses a more generalised notion of inferences, which take the form of natural sentences that preserve the relevance and necessity in relation to the input and output functions required. This has been demonstrated through the Protégé knowledge ontology modelling tool with a Visual Basic plug-in that performs the inference functions.

2.5.2 *Knowledge in Engineering Applications*

Knowledge-Based engineering aims to automate mundane tasks performed by designers by explicitly defining design rules and heuristics used by the designers. This has been implemented through scripts, which define metatasks for each application. These are then used as guidelines in different application scenarios (Gardan and Gardan, 2003).

A review of software projects in this field has been written by Lindvall (2003) who concludes that "*all repository-based software systems support a majority of the phases in the knowledge life cycle, while few systems actually deal with the analysis and synthesis of new knowledge.*"

The use of knowledge in engineering has been illustrated by Becker and Zirpoli (2003) who describe the use of knowledge applied to new product development at FIAT. This has been adversely affected by FIAT's propensity to outsource design

and therefore hollow out its knowledge base. The company is now dependent on systems integrators to provide much of their design ideas as well as their outsourced technology. The work highlights that knowledge needs to be retained for the organisation to have control over what they are producing.

Case based reasoning has been used to aid decision making in new product development (Belecheanu et al, 2003). A search for keywords is performed to retrieve past cases where a similar situation was encountered. Past cases are stored in a central knowledge repository in document form. Any new documentation is also stored in this base. No formal structuring of knowledge is required in this situation.

A method of knowledge representation for computer aided production planning has been published by Grabowik and Knosala (2003). In this approach an object oriented method has been introduced that represents design features as individual objects within a hierarchical class structure. An expert system searches through the knowledge base to find machine tools for particular applications.

Knowledge has also been used to select tools and machines for manufacturing applications (Chung and Peng, 2004). The research presents a clear web-based system, but does not exploit knowledge functions to their full potential. There is no intelligent mapping taking place between the design needs of a project and the properties of different machines and tools that can satisfy those needs.

Although there are several works on the use of knowledge in engineering applications, there is no consistent language that is being used to represent the

knowledge. However a generic knowledge engineering language has been created by Debenham (1998) - see Figure 2-6.

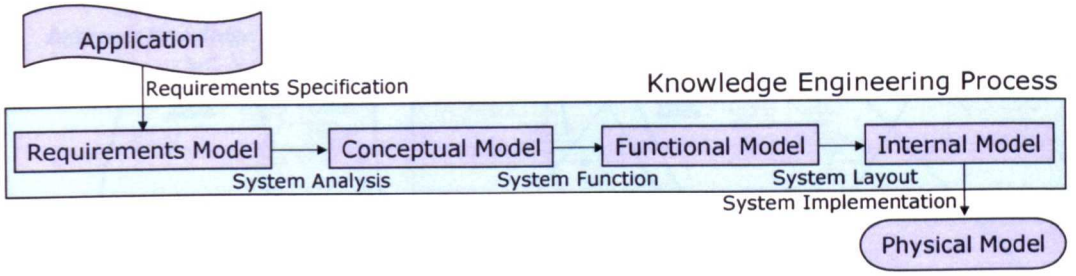


FIGURE 2-6: KNOWLEDGE ENGINEERING APPROACH (*SOURCE: DEBENHAM, 1998*)

Each item in the model is described as data, knowledge or information with relation to a physical or non-physical object. Associations between these are mapped to goals to deliver specific outcomes. The language is described through logic statements and forms a basis for powerful knowledge modelling.

An alternative method is CommonKADS (EU-ESPRIT Project P5258). This research initiative produced a framework for developing knowledge-based applications by Knowledge Acquisition and Document Structuring (KADS) at three levels of abstraction: Domain, Task and Inference. Domain knowledge constitutes all the static data that is required to make decisions, whilst tasks are procedural activities that are needed for the satisfaction of a goal. Inferences are rules that are used by tasks to call upon domain knowledge and to manipulate any data.

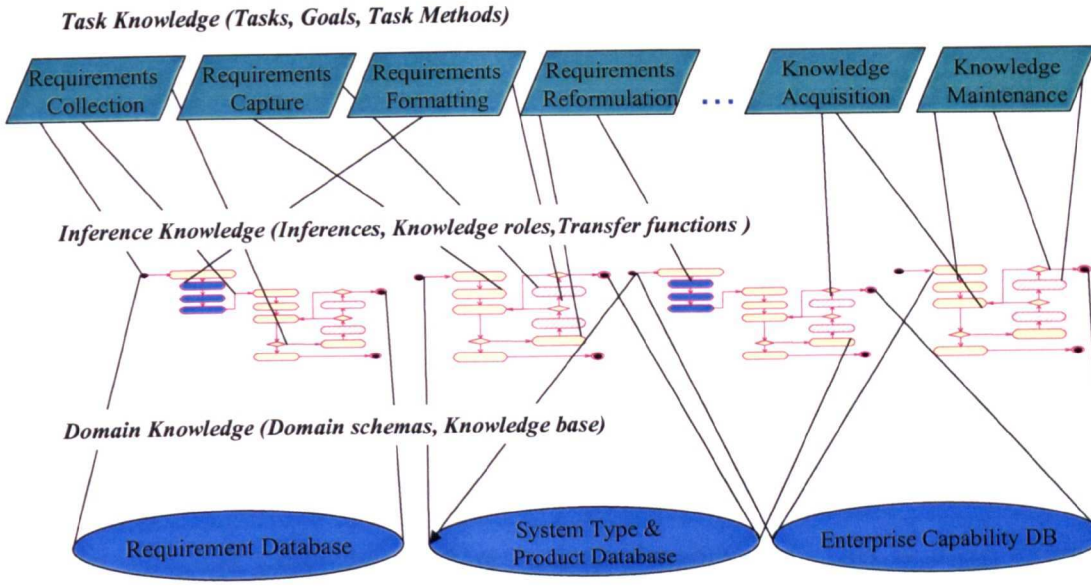


FIGURE 2-7: THE COMMONKADS APPROACH (*SOURCE: ADAPTED FROM SCHREIBER ET AL, 1993*)

The overall approach is described by Schreiber et al (1993). The central theme in the KADS methods is the construction and refinement of models representing the future system, its user(s) and its surroundings. It describes KBS development from two perspectives:

- Result perspective: a set of models, of different aspects of the KBS and its environment, which are continuously improved during a project life-cycle.
- Project management perspective: a risk-driven generic spiral life-cycle model that can be configured into a process adapted to the particular project.

KADS is a product-driven method. This means that progress is defined in terms of products and not in terms of activities. At its simplest and most direct, it offers an established and popular way to document the development of KBSs. When used in full, though, KADS also offers a thorough, methodical approach for developing KBSs.

KADS is a methodology for KBS requirements analysis and design, based on a modelling paradigm. The core concept here comprises a four-layer model of expertise. CommonKADS is the comprehensive guide to KADS that has been developed into a common standard.

KADS prescribes phases; stages and activities; models; documents and deliverables. It provides specialised techniques, project metrics and quality assurance procedures for KBS development. It pays special attention to the special characteristics of KBS and the particular problems inherent in their development. This is done through seven stages as presented by Schreiber (2000):

- Analysis – Analysing the objectives and problems of the client and determining the functional requirements of the prospective KBS. (Output of this phase is the requirements document)

- Design – Design description process consisting of three stages, the functional, behavioural and physical description. Output of this stage is a structure directly supporting the final system artefact (system code), but which still remains implementation independent

- Implementation – When the analysis and design stages are completed, the implementation vehicle can be selected. A choice can be made between programming languages (develop own KBS by starting from scratch), development environments (make own KBS and have some ready-made routines for implementing the component parts) and shells (essentially an empty knowledge base)

- Installation – Installation of a KBS is conducted along the same lines as the installation of a conventional software system (includes further testing of KBS with particular reference to its environment (users, operators, administrators, and other systems)

- Use – When the installation stage has been completed, the system can be used. (This starts familiarising the users with the engine and the system itself)

- Maintenance – This not only involves the maintenance of the KBS but also the maintenance of the inference engine. This happens as soon as the user is no longer satisfied with the system.

- Knowledge refinement – This might be necessary if the experts' knowledge was too costly or impractical to use during the analysis phase. Another reason could be that additional knowledge is required to make the system more complete at a later stage.

The models that are part of the CommonKADS model set are as presented on the commonKADS web site. These models contain information on the four levels of knowledge:

- Goals Knowledge – the goals of the system in terms of what the end user expects to gain from use of the system. This can be directly mapped onto the requirements elicited from the requirements elicitation stage.

- Task knowledge – the goals can be decomposed into sequences and/or hierarchies of tasks. These operations have to be performed in order to satisfy the goal.

- Inference Knowledge – this is knowledge about processes that need to take place to use the fourth type of knowledge (Domain Knowledge) in order to perform tasks. An inference engine may also be required to seek out the relevant static (domain) knowledge fragments for a particular task.

- Domain Knowledge – this is static knowledge that may be stored in databases within an information system until requested or recalled by an inference engine.

The description of the CommonKADS framework reveals the complexity involved in implementation of the framework and CommonKADS has received criticism for this. The framework has been found to be costly to implement due to the detailed documentation required and although this may be still be beneficial for large firms, it may not be appropriate for small or medium-sized organisations.

2.6 KNOWLEDGE GAPS IN THE CURRENT LITERATURE

Knowledge gaps in the research area have been mapped out in the Assembly-Net Roadmap (Onori et. al, 2003). This is an in depth survey of precision assembly technologies in Europe and outlines the research needs for precision assembly to succeed in Europe and Requirements Specification for Reconfigurable Assembly Systems is a big step towards achieving the following recommendations arising from the Assembly-Net Roadmap:

- Form a library of standardised processes
- Initiate assembly process capability studies
- Enhance automation knowledge in all product classes
- Enhance process knowledge at all levels
- Apply methods that link product design to system design and production requirements to product features

Moreover the knowledge gaps addressed in this study are discussed below.

2.6.1 *Limited Formalisation of Assembly Knowledge*

Many studies have been found that report on design for assembly and factors that need to be considered. However, these generally confront issues such as line balancing or assembly at the macro level. However nothing has been found in the literature dealing with description of assembly actions and processes at the detailed level.

Assembly processes and reasons for process selection need to be described in more depth, where the parameters that determine the outcome of each action are defined and formalised.

2.6.2 *Limited Application of Requirements Engineering to Reconfigurable Assembly System Design*

Requirements engineering was originally developed in the defence industry and has been a well established discipline in software engineering. Although the benefits of expanding this phenomenon to assembly system design are clear, this has only been

applied to conventional assembly systems to date and further expansion to Reconfigurable Assembly Systems is essential.

2.6.3 *Limited Exploitation of Knowledge-Based Approaches in Industry*

It is clear from reviewing the developments in knowledge engineering and knowledge based systems that these can be used to consolidate the knowledge of Reconfigurable Assembly System designers. The aim is to provide a basis for system users to supply more accurate requirements specifications, which can be easily turned into system requirements with minimal effort from the designers. Moreover the transition from user requirements to system requirements can be less time consuming for designers by introducing an expert system that makes the process semi-automatic. This will give the designers more time and energy to spend on the creative aspects of system design as they will have a clearer understanding of what they need to produce.

2.7 CHAPTER SUMMARY

The literature review has uncovered knowledge gaps in terms of limited formalisations of assembly knowledge; limited application of requirements engineering to Reconfigurable Assembly System design; and limited exploitation of knowledge-based approaches in industry.

The need for reconfigurable assembly has risen from the inadequacies of flexible assembly where expensive machinery with excess functional capabilities is employed.

A variety of methods have been found for coping with the complexities of assembly system design including heuristic methods based on question and answer approaches and systematic approaches based on concurrent engineering principles.

Requirements engineering is a key stage in the system design process as requirements need to be specified correctly to avoid unnecessary errors, which would be expensive and time consuming to correct later. Elicitation of user requirements and the steps involved in deriving system requirements from user requirements are the key processes that have not been explored sufficiently in the literature.

Knowledge-Based engineering aims to automate mundane tasks performed by designers through explicit definition of design rules and heuristics used by the designers. Rule based systems have been identified as a particularly useful enabling technology to aid requirements specification of reconfigurable assembly systems.

Moreover, the literature review has not discovered sufficient research content for integrating these aspects to form a knowledge based requirements specification methodology for reconfigurable assembly systems.

3 RESEARCH METHODS

3.1 INTRODUCTION

The work described in this thesis began in October 2000 and the original idea was based on merely providing a template for requirements engineering of automated assembly systems. However, this had to be refined as discussions with industry revealed that a more focused application and approach needed to be taken.

Developments in the research area since the beginning of the research period meant that greater attention had to be paid to the flexibility aspects and how to make flexibility more practical and affordable for industry. Moreover the pace of technological development in the field meant that an approach needed to be founded that could be adapted for future needs as and when they changed. Hence it was essential to have good contact with fellow researchers in the field as well as industrial contacts to gain foresight as to what would be required in the future.

Throughout the discussions it was found that industry was lacking a common platform for communication – a common language. The author believes that the research presented in this thesis contributes some solutions to this problem. However, further work needs to be done to make a long term impact on industrial practice.

The theoretical foundation for the research has already been explained in the literature review. However, the purpose of this chapter is to highlight the research methods used to fill the knowledge gaps and explain the decisions made during the research.

An overall research methodology is presented and the various aspects of this methodology are explained in this chapter.

3.2 Overall Research Methodology

The research methodology explains the steps undertaken to derive the research outcomes. It was constructed through outcomes of the literature review and advice from research support.

The aim of the methodology was to provide a structured means to achieving the research objectives highlighted in 1.3. This is illustrated in Figure 3-1

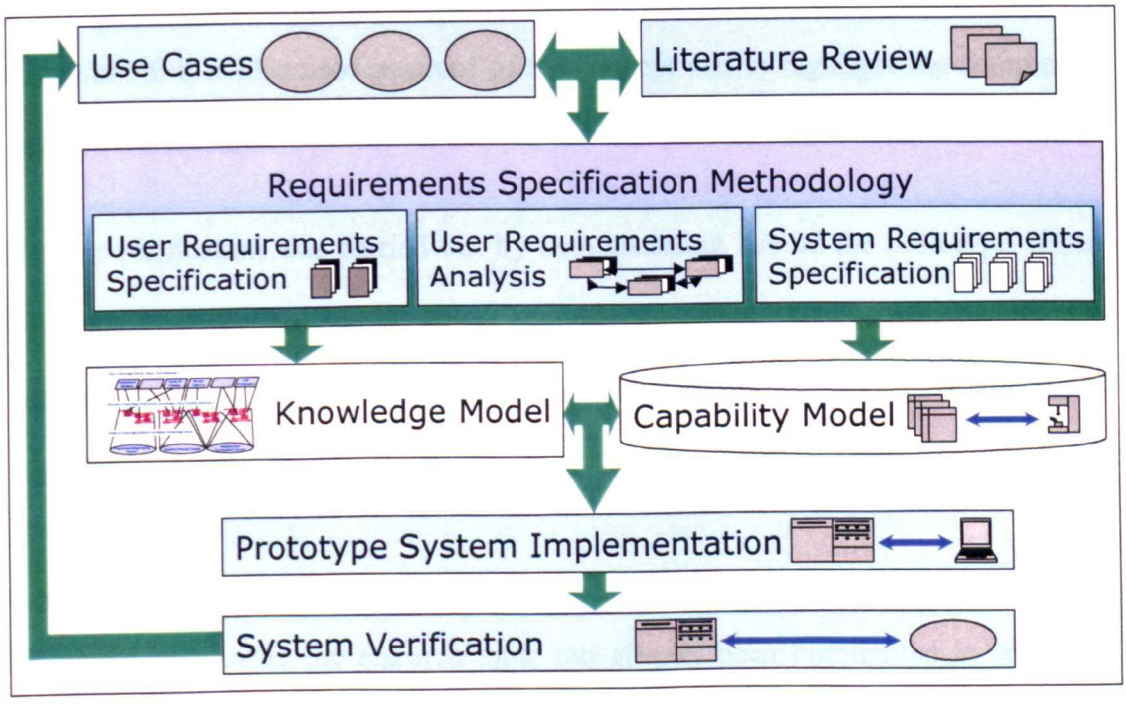


Figure 3-1: Research Methodology for Requirements Specification of Reconfigurable Assembly Systems

The research methodology consists of inputs, processes and outputs where industrial use cases and findings from literature are seen as the main inputs; the development of the requirements specification methodology, knowledge model and capability model

and system verification are the main processes; and the pilot system implementation integrates the research findings to produce the main output.

Findings from the use cases and literature review were used for the parallel development of the requirements specification methodology, knowledge model and capability model. None of these three aspects could be worked on independently as developments in each aspect had a direct impact on the others.

A prototype system was developed to integrate the three development areas and to demonstrate and prove the concept. Demonstration was particularly important in engaging the enthusiasm of industrial contacts to win their trust and encourage their cooperation. It is also used as proof of the concept and to highlight the commercial value of the research.

System verification was carried out by demonstrating use of the prototype system with data from past projects. This was shown to the companies that provided the data and changes were made to the models to reflect any improvements that could be made.

3.3 Literature Review

Literature relevant to the research topic has already been highlighted in section 2. Furthermore, additional sources of information were analysed to populate the knowledge model and to structure the assembly system capability model.

Particular attention was paid to the Manufacturing Assembly Handbook developed by Lotter (1989). Although the classification of assembly system in this text is over a decade old, the structure of assembly systems highlighted in this book has proved to

be relevant particularly in constructing the assembly system capability model. Lotter's work classifies assembly processes in terms of feeding, handling, special operations, testing and joining and presents the characteristics of each.

This was reviewed in conjunction with the Assembly-Net Assembly System Taxonomy (Dini, 2002) and the European Precision Assembly Roadmap 2012 (Onori et al, 2004). Recommendations from these two publications were considered during this research project especially in the construction of the knowledge model and development of the capability model.

3.4 INDUSTRIAL USE CASES

Use cases were captured through interaction with industrial contacts. Each use case is a collection of scenarios faced by the companies involved in the context of assembly system design and development. This was done through the following activities over the period of research:

- Industrial Visits – companies were visited and assembly activities in the respective factories were reviewed and origins of assembly equipment were found. Furthermore people responsible for the deployment of assembly equipment in these factories were asked about the role and level of requirements specification methods.
- Project Shadowing – an assembly system design project was shadowed for 8 months from the requirements specification to the final delivery and installation of the finished assembly system to discover the challenges faced by systems integrators when developing assembly systems. This is the Southco Glove Box Latch Assembly (SCO2) mentioned in the Verification Chapter (See Section 7)

- Project Reviews – project reviews were conducted reflecting on past assembly system design projects. Assembly projects with TQC Ltd, Bosch Rexroth, Southco Ltd and GlaxoSmithKline were examined with the aim of extracting information on their requirements engineering methods and the use of knowledge in these methods.
- Participation at the European Postgraduate Summer School in Precision Assembly – two sessions of this summer school were attended, which presented the opportunity to visit companies outside of the UK to gather data. Further insight into the use of assembly technologies was gained through the summer school. More specifically companies visited were questioned on their application of requirements engineering methods.

Decision making criteria and the experience of participants in these activities was noted and later formed the basis of the requirements specification, knowledge and capability models.

Furthermore, through participation in these activities it was found that people within the companies had their own heuristic methods of dealing with enquiries and interacting with other stakeholders in assembly projects. Although these were established (through experience) in the building of 'normal' or flexible assembly systems, factors to take into account were unclear when it came to designing Reconfigurable Assembly Systems. The result was that modular assembly systems were created, which were not necessarily reconfigurable.

It became clear that further analysis of the need for system reconfiguration and the implications that this would have on an assembly system was needed. Figure 3-2 was created as a result of this analysis.

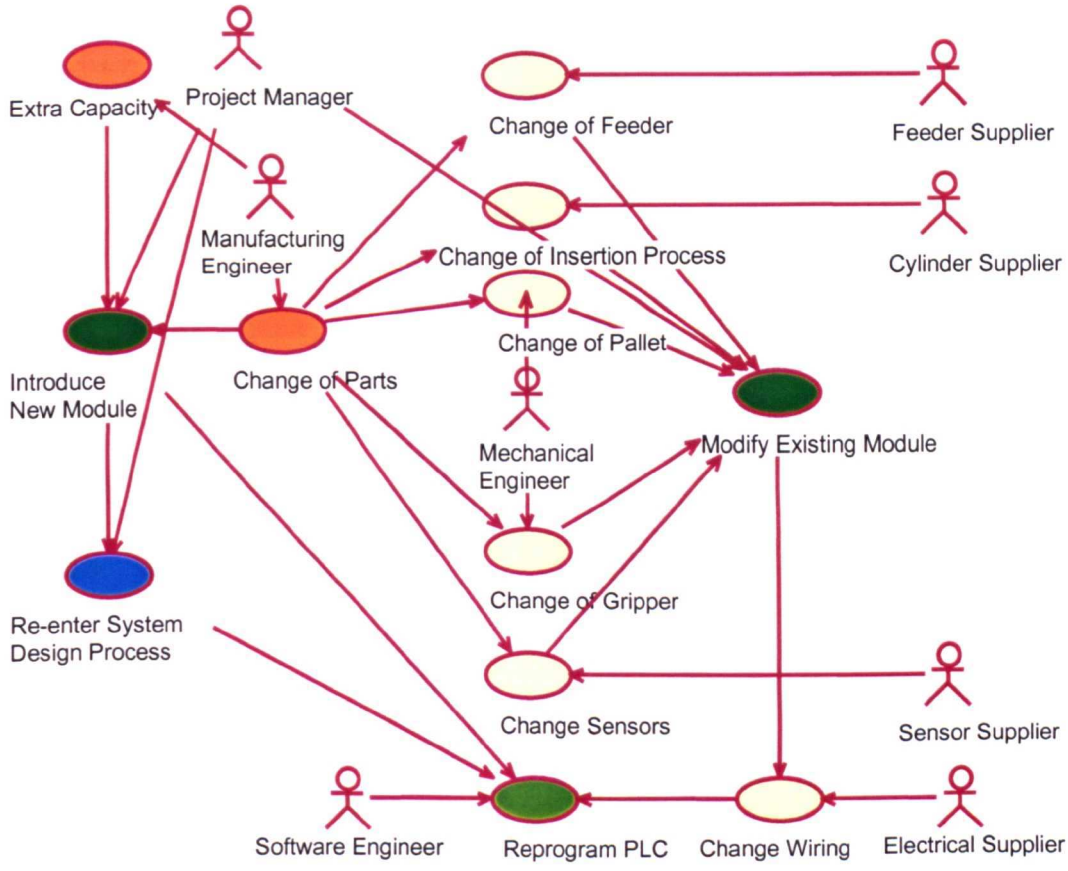


FIGURE 3-2: USE CASE DIAGRAM SHOWING CAUSE AND EFFECT OF SYSTEM RECONFIGURATION

It was found that system reconfiguration was required for one (or both) of two reasons as illustrated in Figure 3-2. There was either a change in part or a need to increase capacity, which was overseen by the manufacturing engineer.

For an increase in capacity it was necessary to introduce a new assembly module and this involved re-entering the assembly system design process, designing and integrating the new module(s) and then reprogramming the control system to adapt to the change. A project manager took ownership of this process. If the change in parts

was significant enough to warrant a new assembly module then that was introduced in a similar way.

However, smaller changes in parts resulted in the modification of existing assembly modules. This was overseen by a project manager, and involved changes to feeders, processes, pallets, gripping devices, sensors and wiring. Each of these formed their own use cases, managed by the respective authority in that field, be it a supplier of equipment or an engineer who had to manufacture the components.

The results from analysis of examples in literature and the industrial use cases formed the basis of the parallel development of the knowledge model, capability model and the requirements specification methodology. It was established at this stage that requirements specification for Reconfigurable Assembly Systems involved:

- specification of business constraints;
- accurate description of the product and its parts;
- understanding how the parts are linked to form the product.

The knowledge model, capability model and requirements specification methodology combine to deliver this. Each of these is explained in the next section.

3.5 Parallel Development of Knowledge Model, Capability Model and Requirements Specification Methodology

Although the requirements specification methodology, knowledge model and capability models are three separate entities developed to satisfy the research objectives, they were developed in parallel. Changes in one model meant that

changes had to be made to the others as they form part of a single process. The development of each aspect is described below.

3.5.1 Requirements Specification Methodology

The construction of the requirements specification methodology began with analysis of the use cases and literature taken from companies. This was then amalgamated and reconstructed to form five use cases in the Unified Modelling Language (UML). The use of UML was recommended as a powerful way of modelling complex interaction and the Rational Unified Process (RUP) was used due to its ability to provide a means of modelling different aspects of the methodology in a variety of views. A simplified diagram of the methodology development is revealed in Figure 3-3.

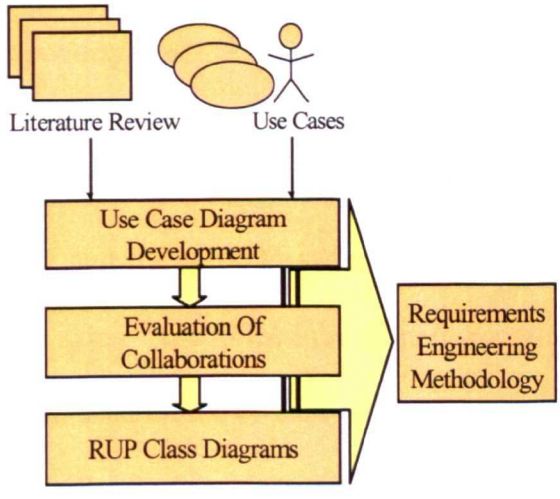


FIGURE 3-3: DEVELOPMENT OF REQUIREMENTS ENGINEERING METHODOLOGY

As part of the RUP a use case diagram was constructed to show these as the main aspects of requirements specification for Reconfigurable Assembly Systems. An important distinction to make here is that between the raw use cases extracted from

industry and the use cases shown in the use case diagrams referred to here, which have been created to summarise the scope of activity taking place in the former. Diagrammatic descriptions of each use case using the UML notation are presented in Appendix A.

Collaboration diagrams were created to model the tasks within each use case in the use case diagram. The advantage of this form of representation is that each task is modelled as a network of sequences and ownership of that task is shown on the same diagram. Collaboration diagrams were then used, as a basis for class diagrams, showing the information needed for execution of the use case.

The result of the use case decomposition and analysis was a set of diagrams to use as a foundation for the knowledge model.

3.5.2 Knowledge Model

The knowledge model was created from information extracted from use cases and literature (Figure 3-4). Task knowledge was defined in the requirements specification methodology and the static knowledge needed to perform each task was extracted from the use cases. This formed the domain knowledge schema and was validated by revisiting the description of each use case*. Thereafter links between the domain knowledge and tasks were investigated to evaluate the domain knowledge needs for the execution of each task. Refinement of these links formed the basis of the

* The complete set of domain knowledge components and their relationships are illustrated in Appendix B

inference rules. Validation was conducted by checking each use case for consistency with the resulting method from the knowledge model. The model was then formalised and could be implemented.

The decision to consider the task structure first, followed by the domain and finally, the inferences was made due to the tacit nature of inference knowledge. Task knowledge was relatively easy to extract from use cases and domain knowledge was derived from the tasks without much difficulty, however inference knowledge was much more complex as some of the decision-making rationale was unclear. Inferences could only be found after further consultation with the industrial contacts that supplied the use cases. This involved asking the designers and engineers to reveal the basis of their decision-making for the projects, which in some cases, they were either reluctant to do or could not explain their rationale in words. A complete set of mapping inference rules extracted from the experts is found in Appendix C.

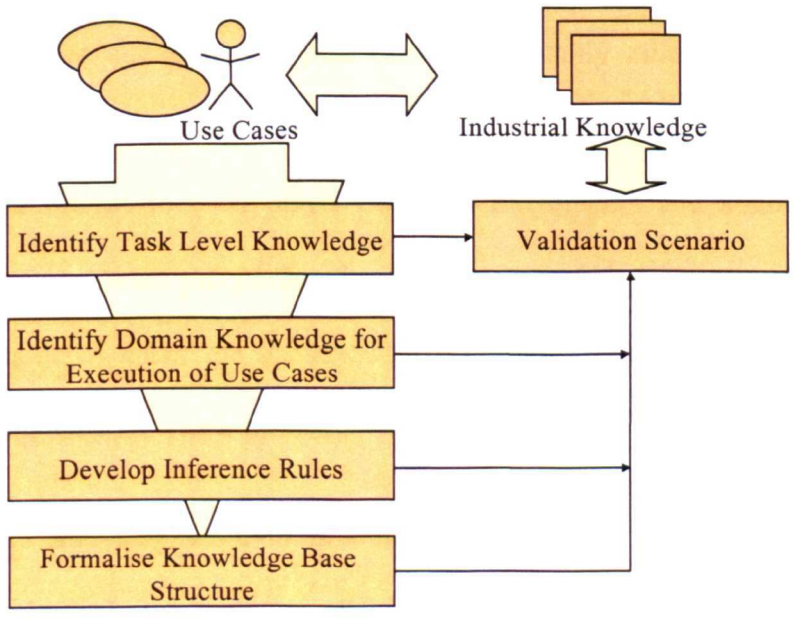


FIGURE 3-4: DEVELOPMENT OF KNOWLEDGE MODEL

The knowledge model was validated by implementing the industrial use cases in the demonstration environment and discussing the implementation with the designers and engineers from industry.

3.5.3 *Capability Model*

The assembly system capability model is a specialised subset of the knowledge model. It forms a major part of Reconfigurable Assembly System development because it charts the functions the system is able to perform, both currently and in the future. Hence future capability can be built within the system even if future applications are not well defined.

The aim of the capability model is to allow the user to accommodate system reconfiguration at the initial design phase. It was constructed after analysing a range of existing assembly systems that were either in operation within the companies visited or in the design phase by systems integrators.

Particular attention was paid on how assembly modules within the system were moving, the assembly actions taking place, functions being performed and how these were related to each other. Common movements and actions to achieve similar functions were of particular interest. Potential for expansion and change was considered under the scenario of the current product being modified or the system being used for a new product.

After analysing this data three levels of actions in the assembly system were found. The logical representation of this finding was within a capability model containing clusters of different actions. A similar approach was found in the field of precision

machining where clusters of form generating schema were used to create resource elements that provided modular functionality for different machining tasks (Ratchev et.al, 2000). Some of the principles were applied to create the assembly system capability model. It was found that different actions could be clustered to represent certain capabilities or movements common to different operations.

When actions were grouped in this way it was possible to model a level of modularity which could be introduced to an assembly system as previously hidden commonalities became easier to identify. Some assumptions form the basis of this model:

- That assembly modules are interchangeable and integratable
- Modules have common interfaces for electronics, mechanics and control.

The precept for the capability model is that firms will invest in this type of development and that technology will become applicable because systems will improve. This work will improve the use of reconfigurable technology by providing a medium to promote and extend current capability and untapped potential.

3.6 IMPLEMENTATION AND VERIFICATION

So far in this chapter the input to the research and reasoning behind the parallel development of the requirements engineering methodology, knowledge model, and the capability model has been addressed. This has been integrated and implemented in a pilot environment, which demonstrates the research output.

3.6.1 Data Model Implementation

The data model was implemented using the MySQL database software. The structure was designed using the PowerDesigner tool and this was transported into the MySQL framework. This was chosen as the data structure had to be easy to update and include new assembly technology when it became available. MySQL was particularly suitable as it is a widely used freeware, minimising legal and intellectual property difficulties. For the purposes of this study easy import and export of data was essential as well as creation, storage and editing of data. The software was tested and it was found that its functionality matched research and implementation needs. The software had been tried and tested for other applications in the research group and was found to be easy to use, functional and reliable.

3.6.2 Knowledge Model Implementation

Expert systems are a common way of implementing knowledge functions that are needed for knowledge models to work. However, the research aims move beyond this so a knowledge system that was compatible with industry databases was needed. The requirements for the expert system were that it would be easy to model and update, as new knowledge became part of the overall package. The system had to meet industry needs so that companies could manage their own knowledge by using this shell as a plug-in. The aim was to provide a pool of system creation knowledge accessed by companies and implemented in their own context in a confidential environment.

The CLIPS expert system shell was used to demonstrate how knowledge could be applied to reason with data stored in a data structure. However further attempts with

this were abandoned in favour of the Java capabilities of JESS. Furthermore, a JESS plug-in to the protégé domain modelling tool was available, which meant that all the knowledge could be integrated into a single application. Further experimentation resulted in the discarding of protégé as the software was too time consuming to use and updating knowledge proved to be difficult.

3.6.3 *Methodology Implementation*

The methodology was integrated through a user interface programmed in Java. This was defined and developed through interaction with industry, where systems integrators suggested improvements that could be made to several versions that were implemented. User roles were separated through the definition of different user types and the functionality of the system was determined according to each user type. For example, system users could participate in only the system user tasks and so forth.

The main reason for using Java was the object oriented structure and platform independent nature of the language. This meant that updates could be implemented easily and that if the system was developed into a commercial web based environment then it could be used by anyone irrespective of their IT system. Java is an established web friendly language so this was a positive factor in choosing it for the research test environment.

3.6.4 *Verification with Experts from Industry and Use Cases*

One of the research visions was to make the results relevant to industry and that meant consulting industry at each stage of development. The result was iterative improvements that were made to the original idea. Mostly this was in the

implementation where modifications were made to the test environment and data instances were added to the knowledge domain.

The research has been presented at various conferences (Ratchev and Hirani, 2001a, 2001b), (Hirani and Ratchev, 2002a, 2002b, 2003), (Lohse et al., 2003), (Hirani et al., 2004), (Ratchev et. al, 2004), which has resulted in generation of both academic and industrial interest. Feedback from the presentations at these conferences was used to make further modifications to the research and to give the research more meaning.

Presentations at these conferences involved the demonstration of the research through implementation of the use cases. The system was tested taking data from the industrial use cases and additional functions were added as a result. These were also presented to engineers and designers to verify the findings with experts in the field.

3.7 CHAPTER SUMMARY

It is proposed that analysis of knowledge extracted from industrial use cases and literature will form a starting point for developing a Requirements Engineering Model and Methodology. This will be supported by an Assembly System Capability Model, which will provide an indication of functional capabilities of assembly operations and a Knowledge Model that will capture the rules and facts needed for requirements specification of Reconfigurable Assembly Systems. A Pilot Environment will be created for Demonstration of Requirements Specification of one of a kind Reconfigurable Assembly Systems.

The system will be applied at three levels where task level knowledge will be implemented through object oriented programs, inference knowledge will be created and executed with an inference engine and domain knowledge will be managed by a database. The integration of these three technologies will provide the necessary functionality for requirements specification of one of a kind reconfigurable assembly systems. It is proposed that the work will be verified by application to an industrial case study within a pilot environment.

4 THE ASSEMBLY SYSTEM REQUIREMENTS SPECIFICATION METHODOLOGY

4.1 INTRODUCTION

The assembly system requirements specification methodology has been developed to shorten assembly system development lead times and to provide more accurate requirements specifications which system integrators can use to develop reconfigurable assembly systems. Moreover the approach will lead to more efficient development as expensive design rework is avoided by the provision of structured and relevant requirements specified to the correct level of detail.

The methodology covers the interaction between the systems integrator, who integrates the various assembly technologies available to satisfy the needs of the system user, who is going to use the assembly system to assemble one or many products. The system user defines a set of requirements, which are processed by the system integrator to define the functional specification of the assembly system needed.

Furthermore, the requirements specification process is an integral part of systems engineering (see Figure 4-1). It defines the interaction between the product design cycle and system design cycle.

Although this may be elaborated at a later stage to derive conceptual design and detailed design specification for the assembly system, the scope of this research encompasses only the specification of system requirements.

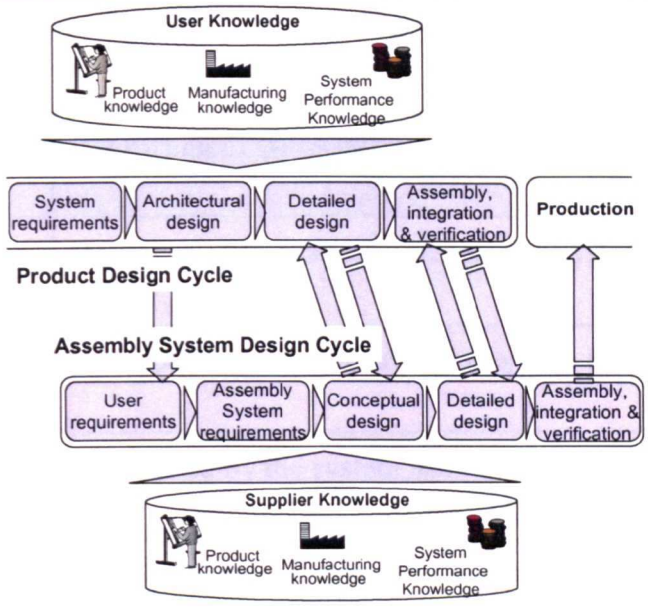


FIGURE 4-1: CONCURRENT REQUIREMENTS SPECIFICATION AND CONCEPTUAL DESIGN OF ASSEMBLY CELLS (*SOURCE: RATCHEV AND HIRANI, 2001*)

For Reconfigurable Assembly Systems this means defining a methodology that encompasses requirements specification for both new Reconfigurable Assembly Systems and for reconfigurations to existing Reconfigurable Assembly Systems.

For the purpose of this study, a requirement is a statement describing a characteristic of the system at an abstract level. Requirements are observed through various angles depending on the nature of the statement. The description of requirement types in Table 4-1 has been constructed to clarify the use of terminology in the present research.

Requirement Type	Requirement Description
Functional	Describes a kinematical function of a system
Non-Functional	Describes a characteristic of the system that is not a function
User	Is specified by the system user
System	Is specified by the system integrator
Current	Is needed immediately
Future	Must be accommodated sometime in the future

TABLE 4-1: REQUIREMENT TYPES

Moreover the system user defines a set of user requirements based on current needs. These are turned into system requirements by the system integrator. The black box that is between these two stages is clarified in the remainder of this chapter. This includes a description of the process, the inputs and outputs, definition and analysis of user requirements, development of system requirements, requirements negotiation and verification as well as an understanding of current and future requirements. This is performed for specification of both new Reconfigurable Assembly Systems and for reconfigurations to existing Reconfigurable Assembly Systems.

4.2 The Requirements Specification Process for Reconfigurable Assembly Systems

The requirements specification process presented here has been developed through extraction of use cases from interaction with industrial contacts. In the requirements domain there are two stakeholders: the system user and the system integrator. They interact through the five use cases as shown in Figure 4-2.

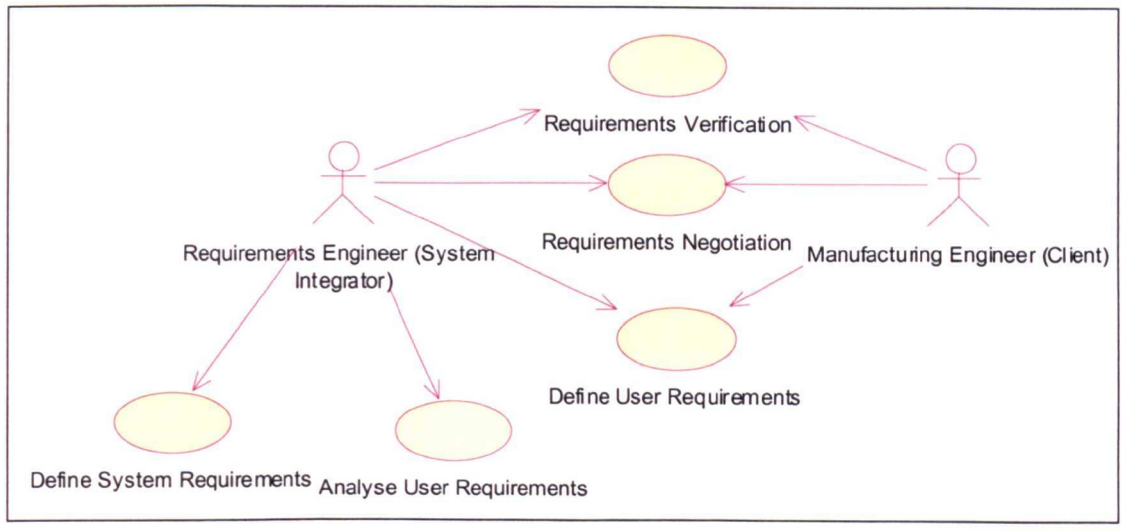


Figure 4-2: Requirements Specification Use Cases

Before explaining the execution of each use case and the scenarios that they cover, it is important to realise that both the system user and the system integrator interact with each other through the use cases. The arrows on the diagram indicate the use cases that belong to each of the Actors. Each use case has a defined starting point with defined inputs and a defined end point (see Appendix A). The purpose of the use cases in this instance is to generate a set of user requirements and then transfer these into system requirements. The system user constructs the user requirements as illustrated in Figure 4-3.

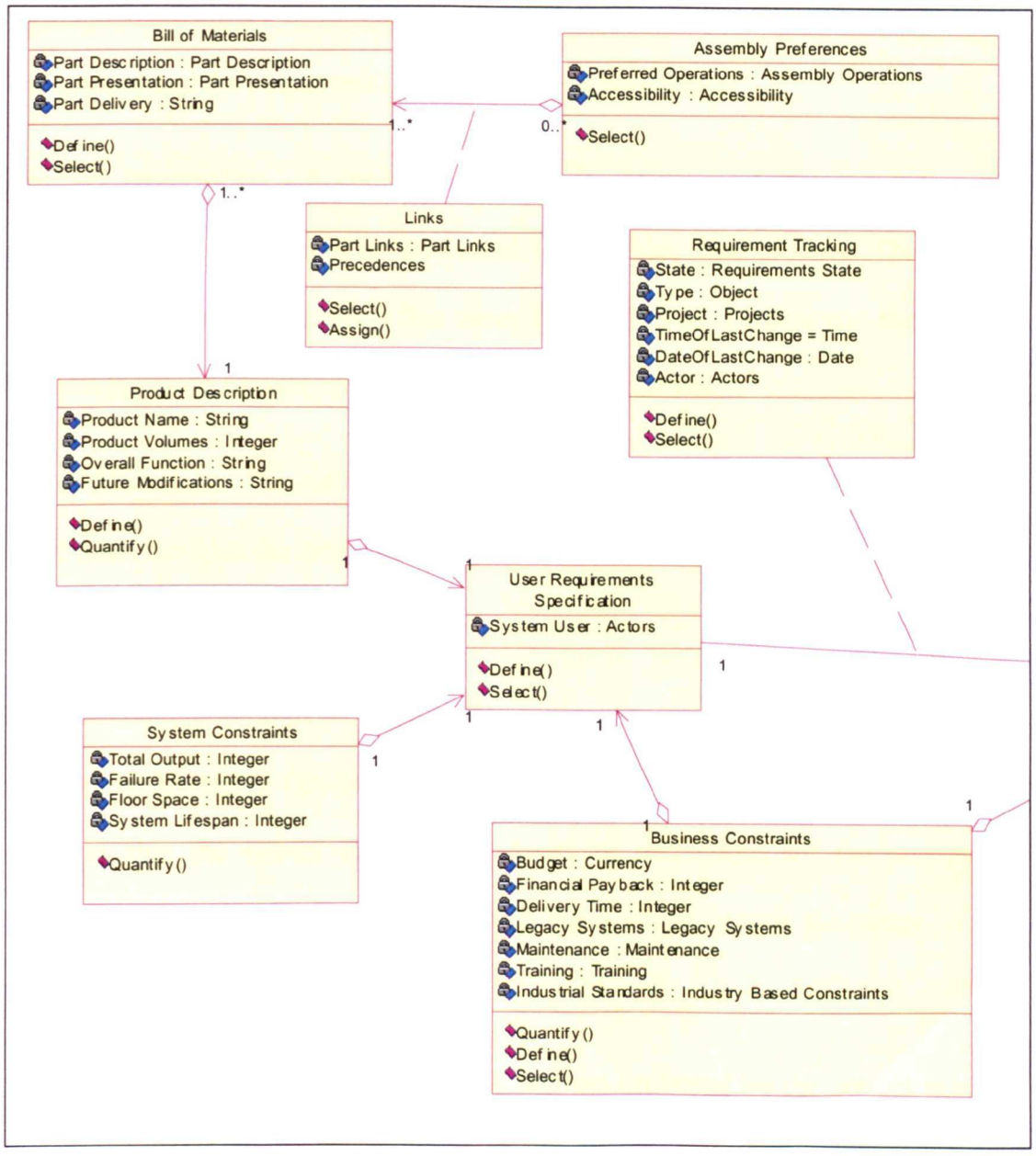


FIGURE 4-3: USER REQUIREMENTS DOCUMENT CLASSES

The user requirements specification includes:

Business Constraints - These outline the business conditions that the project must meet. They include the financial terms and maintenance and training requirements along with relevant information regarding the companies own expertise (legacy systems) that should be taken into account when the assembly solution is developed.

System Constraints - These state the physical dimensions that the assembly solution will occupy and the output that it will be required to produce.

Product Description - A comprehensive profile of the product is needed so that suitable technologies could be employed to assemble that product. The product description includes the aspects that cover the product as a whole and the possible variants that pose a scope for future system reconfiguration.

Bill of Materials - The individual parts that make up the product are described. The description includes the method part supply, as this is provides a basis for developing part handling and feeding systems.

Links - The relationships between different parts are described. This is the basis for developing Assembly Operations at a later stage. The precedence constraints are listed so that the system integrator is aware of restrictions for the purpose of assembly sequencing and planning, which takes place outside the scope of this research.

Assembly Preferences - Any operations that are preferred by the system user are stated so that these can be incorporated into the final design solution.

Use of the classes is described through the description of the knowledge model as explained in Section 5.2

It is the system user's responsibility to provide the required information to the system integrator. The system integrator receives the user requirements and tries to convert

these into system requirements. A class description of the user requirements is shown in Figure 4-4.

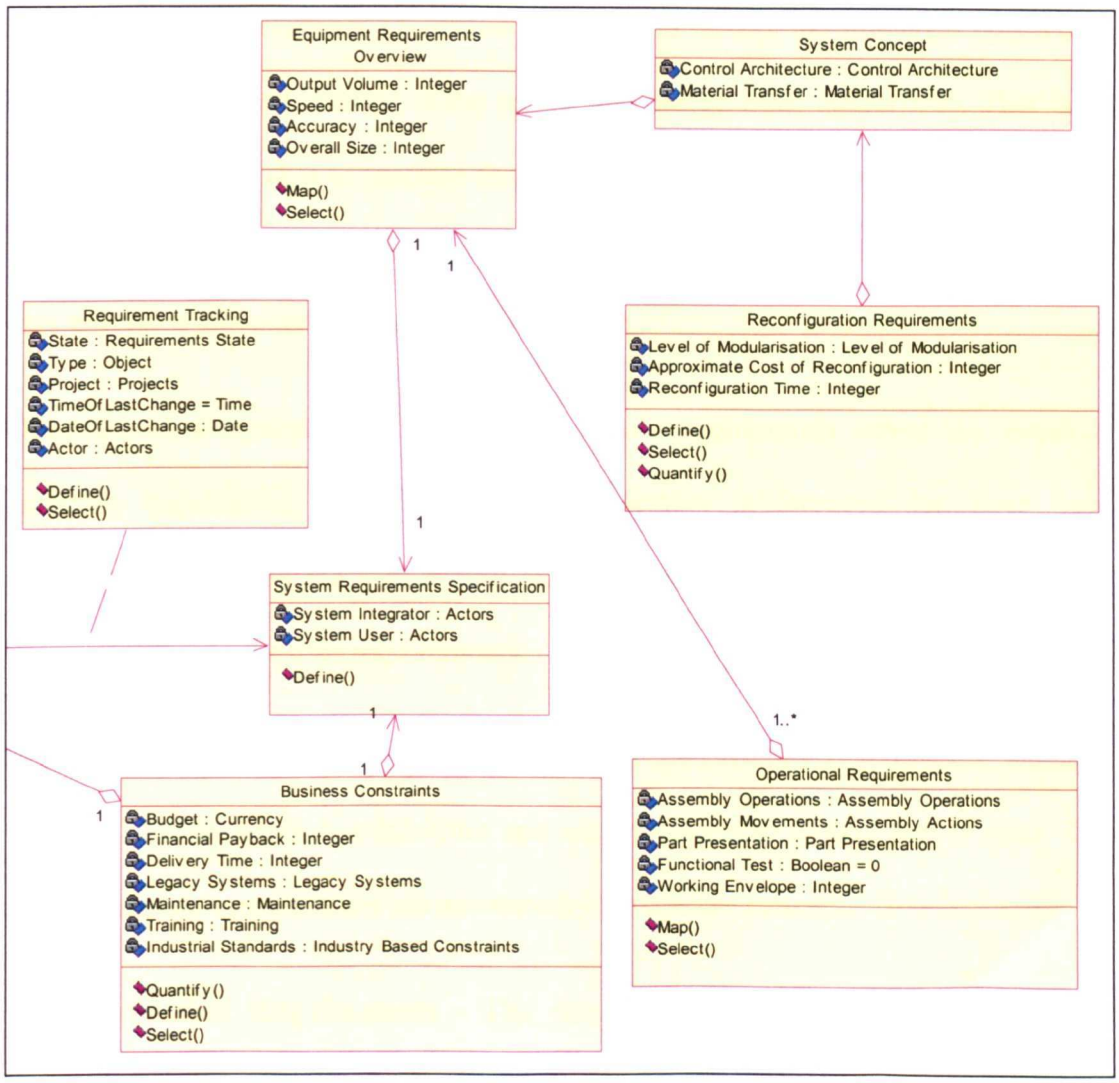


FIGURE 4-4: SYSTEM REQUIREMENTS DOCUMENT CLASSES

The system requirements specification is an amalgamation of classes that are centred on developing system requirements from the user requirements that are specified by the system user. They are processed by system integrator according to the classes described below:

Business Constraints - These remain in their original shape as defined by the system user in the user requirements document.

Equipment Requirements Overview - This class covers the universal constraints that all the equipment defined in the conceptual design must conform to. That is all equipment must be able to produce the required output at the acceptable failure rate. It must also fit into the overall size defined by the user as this is the space that system user will have allocated for the assembly system.

Operational Requirements - The operational requirements cover the required assembly capabilities. This includes any assembly preferences that have been previously defined by the system user and assembly operations that enable the part links described in the user requirements.

System Concept - This is a general overview of the system characteristics, which is described by the control architecture and material transfer system that is used. The reconfiguration requirements are inherited by the system concept.

Reconfiguration Requirements - The reconfiguration requirements describe the level of reconfiguration and the potential time and costs involved. This is based on the lifespan of the system and the number of variants expected to be produced over the system lifespan. A detailed description of this aspect is presented in section 6.6.

Requirement Tracking - Requirements evolve during a project and their status and any changes made to them needs to be logged. The reasons for the change are noted and accepted requirements are marked. The aim of this class is to ensure transparency and traceability in the requirements specification process.

In essence the system user prepares a user requirements specification document and the system integrator produces a system requirements document in response. The system requirements document is verified by the system user and is then used to select equipment for assembly of the product. The relationship between the two documents and their content is illustrated in Figure 4-5.

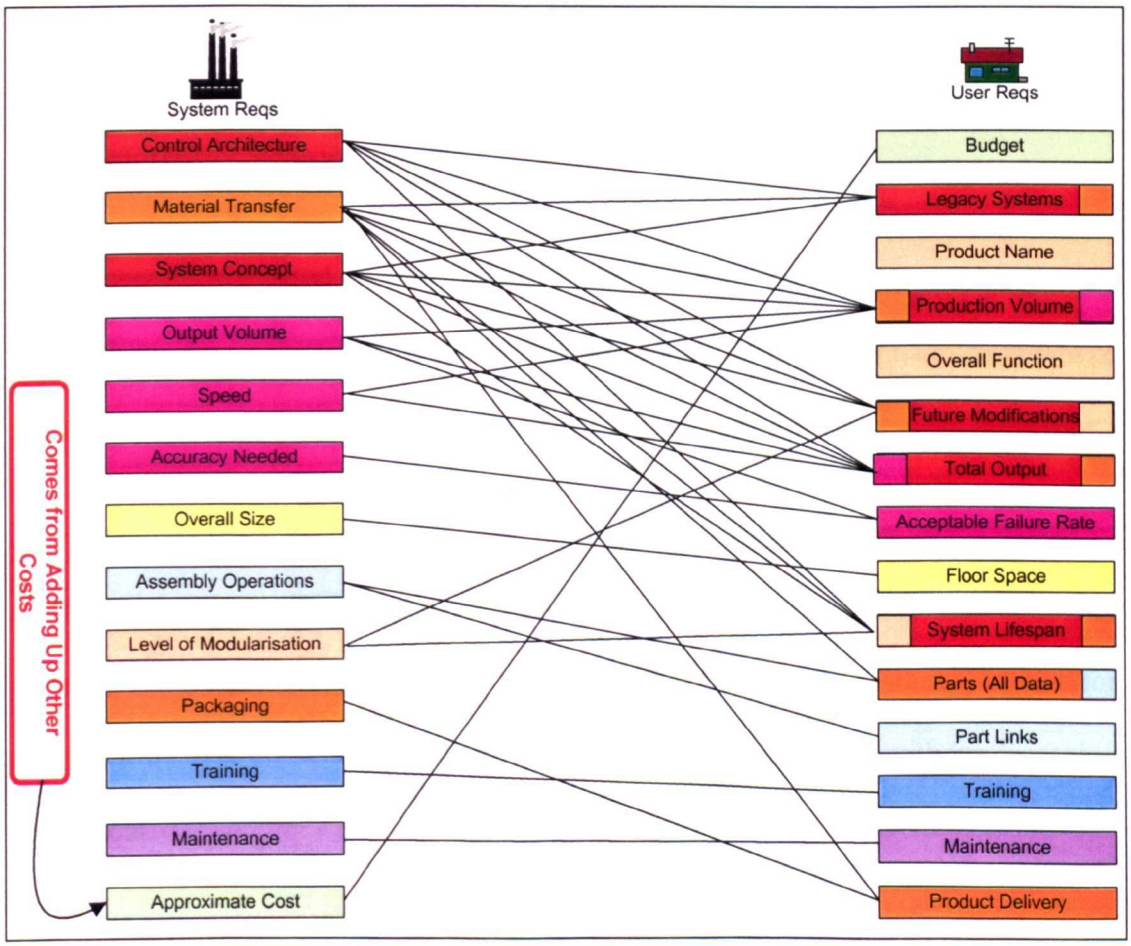


FIGURE 4-5: MAPPING OF USER REQUIREMENTS TO SYSTEM REQUIREMENTS

In Figure 4-5 system requirements categories are shown on the left hand side of the diagram and user requirements are shown on the right hand side. The lines between the categories demonstrate the relationships between the two. From this it can be seen that many user requirements contribute to the development of one system

requirement and vice versa. The complexity involved in this mapping illustrates why requirements specification is a complex process that is seen more as an art than a science.

Through the industrial use case gathering activities (see section 3.4), it has been found that current practice in industry is disorganised and specifications are presented to system integrators at varying levels of detail resulting in vital information being either difficult to find and extract, specified to an unsatisfactory level of detail or missing.

The research presented in this thesis structures the information exchange and establishes a firm agreement on the acceptable level of detail through development of requirements templates, which are filled in by the system user and system integrator.

The requirements template developed in the project includes the specification of future requirements. This serves as a guide to system integrators so that assembly systems can be developed with the prospect of future modifications planned for making it quicker and cheaper to perform system reconfigurations without damaging equipment.

The nature of products that the system user is likely to assemble over the system lifespan is also an issue. If the products are fundamentally different then system reconfiguration will not be practical. However similar products that require similar assembly processes can be catered for through system reconfiguration. Largely this depends on the level of reconfigurability and this is discussed in detail in section 6.6.

Of the use cases presented in Figure 4-2, only the content of the Define User Requirements and Define System Requirements use cases are dependent on the context, i.e., whether a new Reconfigurable Assembly Systems is being defined to whether an existing Reconfigurable Assembly System is being reconfigured.

The user requirements definition covers the system user's expectations of the system. A decision tree charting the general methodology is illustrated in Figure 4-6.

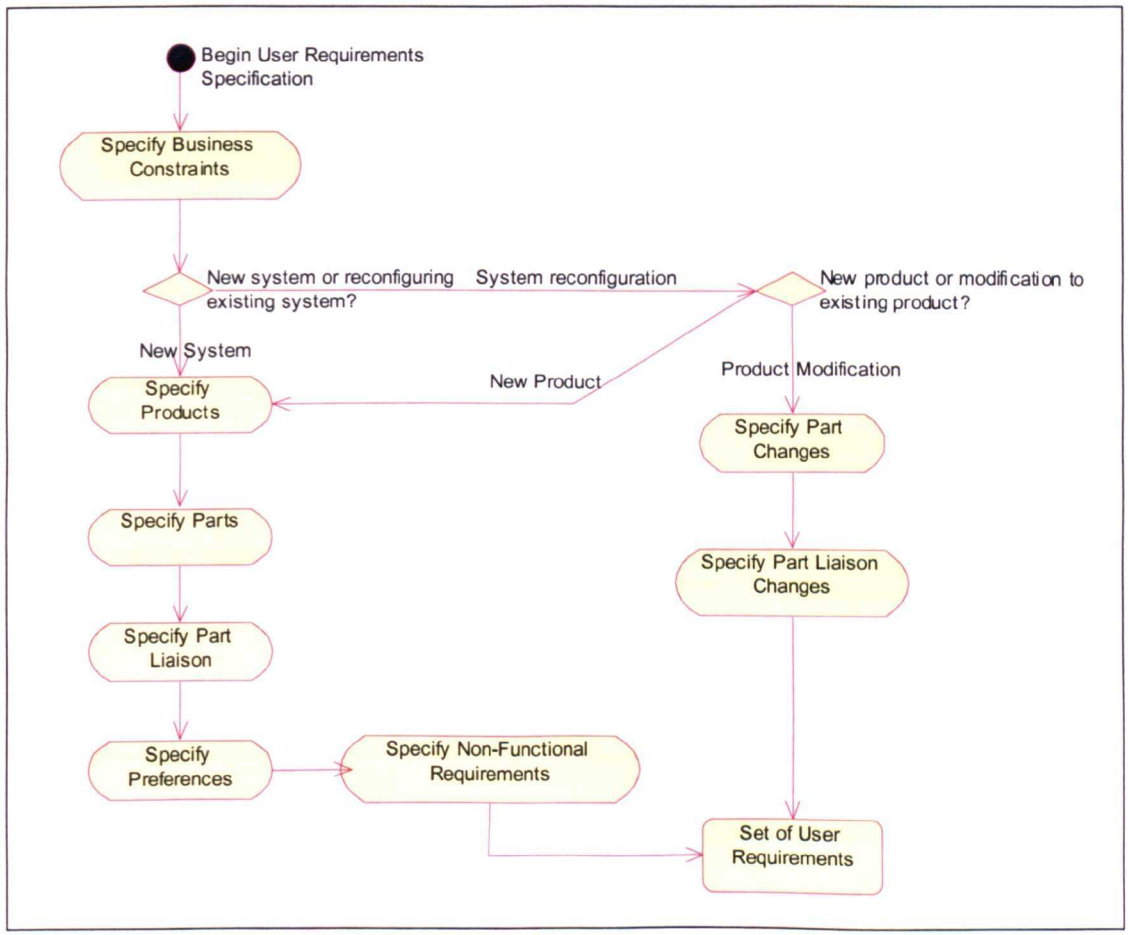


FIGURE 4-6: DECISION TREE FOR USER REQUIREMENTS SPECIFICATION

The user requirements specification follows two different paths, depending on whether the requirements being specified are for a new Reconfigurable Assembly System or for a reconfiguration to an existing assembly system. If a new

Reconfigurable Assembly System is being defined, then all the user requirements need to be specified as is the case if a new product is being introduced for an existing Reconfigurable Assembly System to assemble. However, for reconfiguration due to modifications to existing products, only the parts changes and part liaison changes need to be specified together with the business requirements.

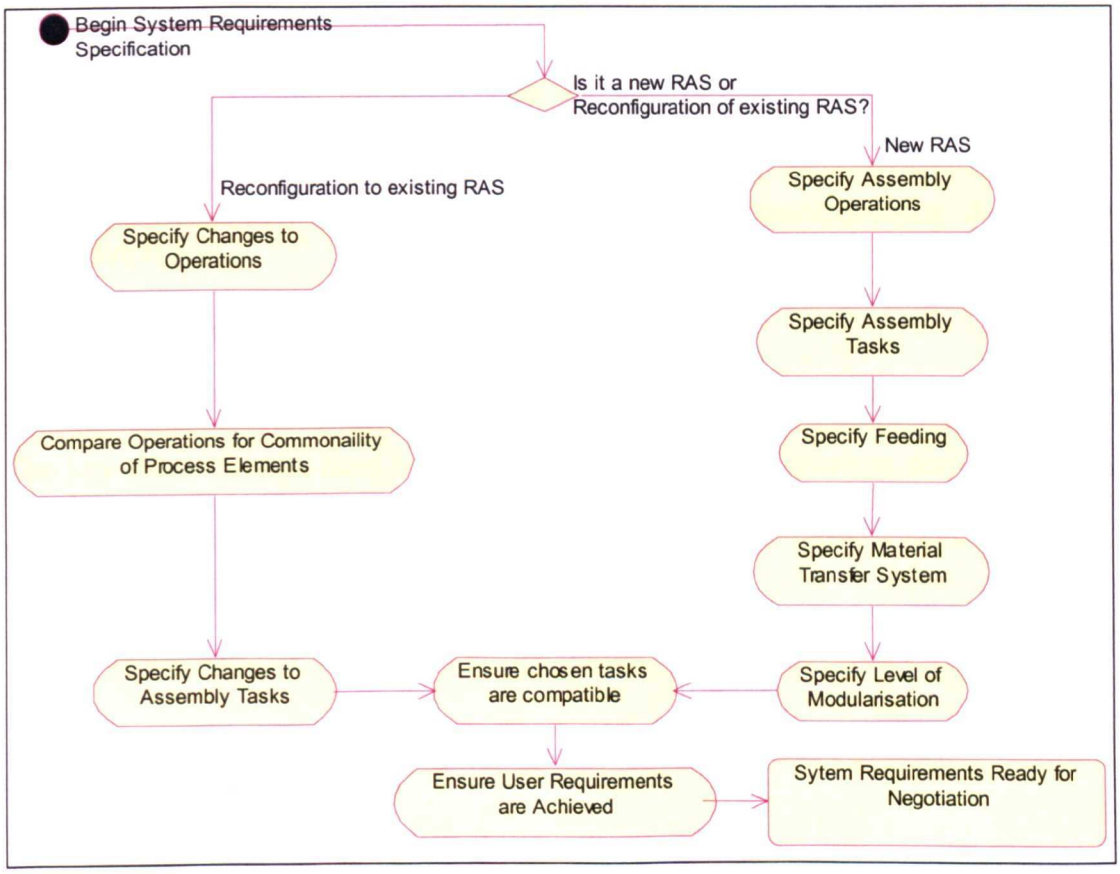


FIGURE 4-7: DECISION TREE FOR SYSTEM REQUIREMENTS SPECIFICATION

Figure 4-7 shows the decision tree for the system requirements specification use case. This follows two distinct paths depending on whether there is a new Reconfigurable Assembly System being specified or reconfigurations are being proposed for an existing assembly system.

For a new Reconfigurable Assembly System, assembly operations, tasks, feeding and material transfer systems need to be specified together with the required level of modularisation. However, only changes to the already specified system requirements need to be specified for system reconfigurations.

Class diagrams have been created to define the detailed task level processes that need to be carried out for these two paths of requirements specification and these are described in the remainder of this chapter.

4.3 Requirements Specification for New Reconfigurable Assembly Systems

4.3.1 User Requirements Definition

The user requirements definition covers the system user's expectations of the system. The key characteristics have been captured and these are shown in sequence in the collaboration diagram Figure 4-8.

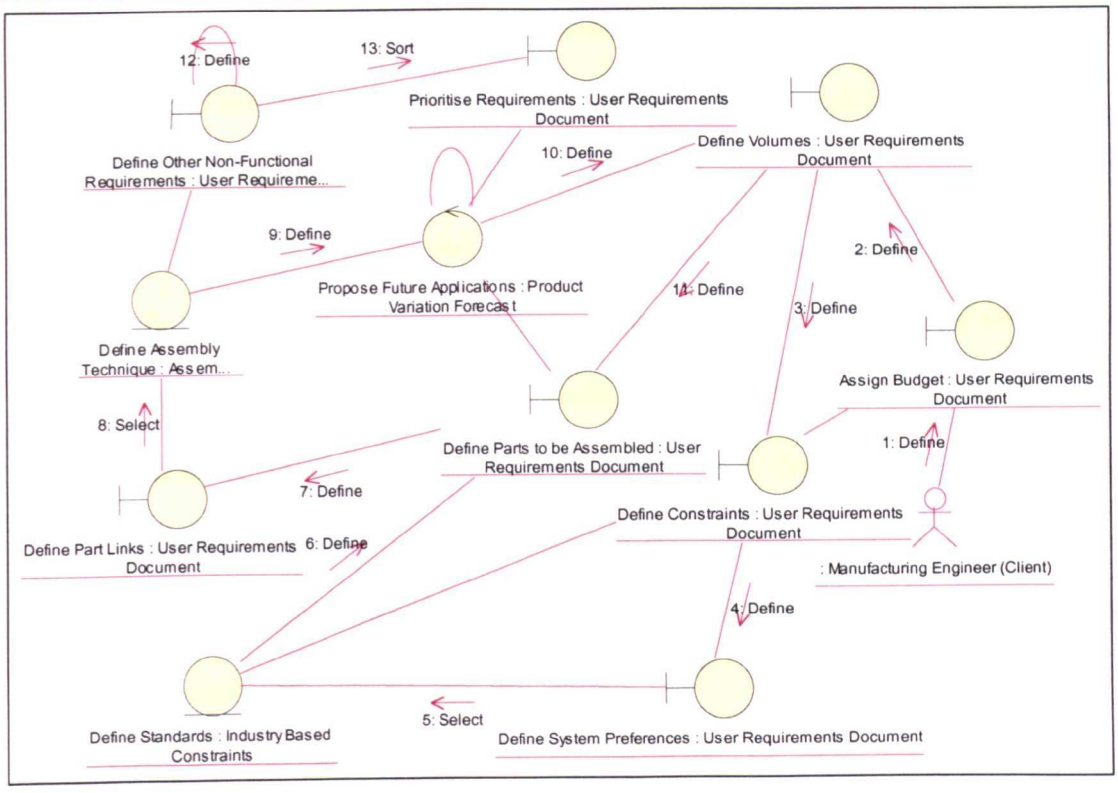


FIGURE 4-8: ACTIVITIES IN USER REQUIREMENTS DEFINITION

The steps depicted by the collaboration diagram involve the system user defining the key characteristics of an assembly system. These are as described below:

I. **Assign Budget:** a financial budget is assigned to the project as a nominal amount that the company is willing to pay for the services of the system integrator in satisfying the requirements of the project. This is the maximum value and one of two key attributes that system integrators use as a constraint when making decisions regarding assembly solutions

II. **Define Volumes:** this is the number of good parts that must be assembled by the assembly system provided by the system integrator. Volume requirements are the second key attribute that system integrators consider when developing assembly systems. Note that this is the number of good parts and parts that are likely to be bad are omitted from this total.

III. **Define Constraints:** the constraints within which the assembly system must operate are stated. This includes failure rates, physical limitations on available floorspace in the factory and other resources, such as air, power and water, which may be needed for the assembly system to operate.

IV. **Define preferences:** the preferences covered here are related to aspects of the assembly system that would be preferred by the system user as they have experience and competence in working with them. For example the company may have staff that are experienced at programming control systems in a windows environment and may want to enforce the system integrator to use a windows based operating environment.

V. **Define standards:** industrial standards that must be adhered to are stated. For example some applications may involve use in a clean room environment. These aspects have to be included at the specification stage so that provisions are made within the system design to provide the necessary functionality.

VI. **Define parts to be assembled:** the nature of the parts that the assembly system must assemble is described so that the correct methods are chosen for the assembly of the product. For example magnetic grippers cannot manipulate non-ferrous metals.

VII. **Define Part Links:** the links between the different parts are specified such that assembly techniques that are suitable for the required link (or part mating) are chosen by the system integrator at a later stage.

VIII. **Suggest assembly technique:** this data does not have to be specified if the system user has no assembly preferences, but is included as some system users have prior experience of using specific assembly techniques. Although

the system integrator does not have to adopt the preferred method it is generally desirable.

IX. **Propose future applications:** in some cases future applications to be performed by the assembly system are known. These are likely to involve some element of system reconfiguration. The type of reconfiguration depends how different the new application is in comparison to the old application and this could be accommodated to make future system reconfiguration easier and faster if initially specified. The prospect of reconfiguration is covered in more depth in 6.6. If parts descriptions and required volumes are available for these then they are described as above.

X. **Define other non-functional requirements:** the other non-functional requirements are concerned with the project details. This includes payment conditions, maintenance and training requirements and system delivery dates and methods.

XI. **Prioritise requirements:** once specified, the requirements are assigned priorities. This depends on which requirements are negotiable and which are most desirable. There are natural trade-offs such as price-quality and speed-reliability and priorities regarding these are established.

All the above activities come together to form the user requirements document. At this stage the user requirements document is solely the product of the system user. This provides a basis from which the system integrator can develop system requirements. The first step in this activity is requirements analysis.

4.3.2 System Requirements Definition

Once the user requirements have been stated they are processed as shown in Figure 4-9. Although the process is a complex one, only the simple case is considered in this chapter. A detailed description is presented in section 5.4 where the relationship between user and system requirements and the knowledge that underpins the decision-making and mapping is explained.

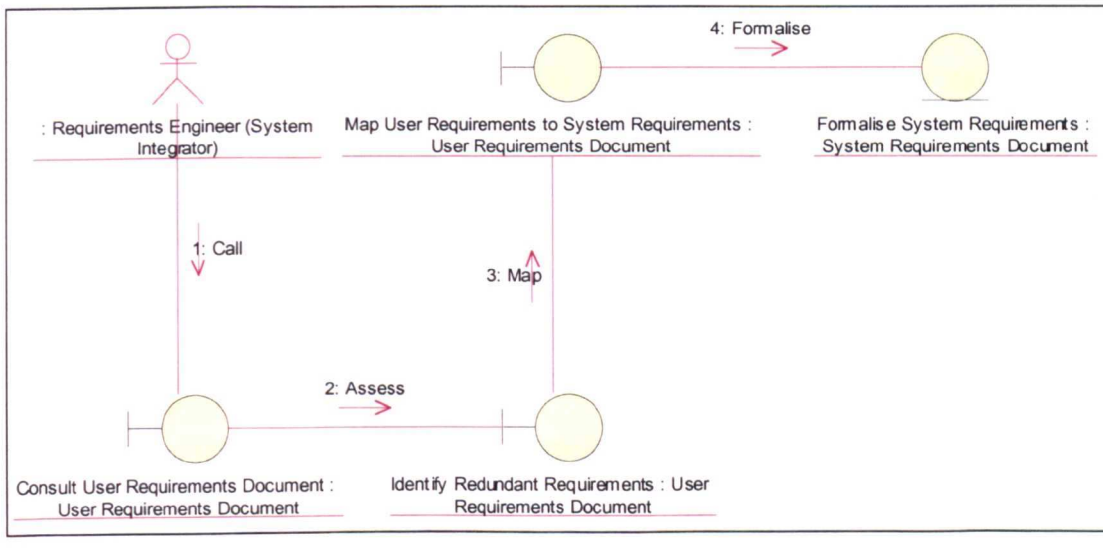


FIGURE 4-9: COLLABORATION DIAGRAM FOR SYSTEM REQUIREMENTS DEFINITION

The completed user requirements document after requirements negotiation serves as the input to the system requirements definition process. The sequence of events is described below:

I. **Consult user requirements document:** the user requirements document is loaded and the various categories of user requirements are presented to the system integrator such that they can make decisions with all the relevant knowledge in front of them. An example of this is presented in section 7.2.

II. **Identify redundant requirements:** any redundant requirements are marked and are ignored for the rest of the requirements specification process. However these are not totally discarded as they may become relevant later.

III. **Map user requirements to system requirements:** user requirements are mapped to system requirements as per the relationship diagram below. Knowledge is used to determine the relationships between requirements and in the most part assembly system requirements are determined automatically using the knowledge stored in the knowledge base. This has already been explained in section 4.2. Any conflicting or missing requirements found at this stage are returned back to the requirements negotiation process for resolution.

IV. **Formalise system requirements:** the formalisation of system requirements is needed to prepare system requirements for transformation into conceptual design. Although conceptual design of assembly systems is outside the scope of this research, it is imperative that system requirements are appropriate for conceptual design as the system requirements document, which is the output from the requirements specification phase serves as the primary input for conceptual design. Once the requirements have been approved they are marked formally.

The formalised system requirements document is the output of the system requirements definition process. The complete document is printed and processed for conceptual design after requirements verification.

4.4 Requirements Specification for Reconfiguration of Existing Assembly Systems

Although Reconfigurable Assembly Systems require a higher initial investment, this can be recouped through long term savings in terms of costs and time, which are a result of cheap and easy system reconfiguration.

Cost savings result as system reconfiguration means there is no need to go through the process to find a complete dedicated system again. Instead modules in the current system can be modified or new modules can be added at a fraction of the cost.

Time savings come from quick integration of modules into a system as opposed to developing a whole system from scratch every time there is a design change. Moreover, time is spent on developing new modules that can be plugged into the system to provide additional functionality.

4.4.1 User Requirements Definition

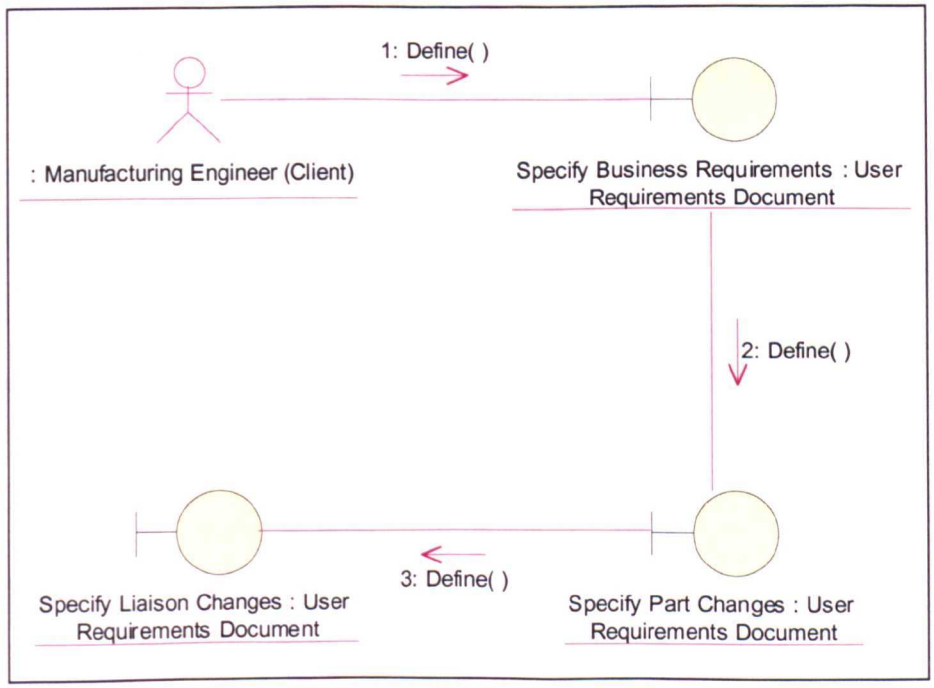


Figure 4-10: Tasks for User Requirements Definition for System Reconfiguration

The user requirements specification for system reconfiguration consists of the specification of part changes and part liaison changes together with the business requirements as described below:

I. **Specify Business Requirements:** Business requirements are specified as the terms of the agreement between the manufacturing engineer (client) and the systems integrator (supplier). This includes the assignment of a budget, definition of product volumes, assembly preferences, design constraints and industrial standards to meet.

II. **Specify Part Changes:** the data for existing parts is retrieved and changes to the part properties are made to coincide with the design changes made to the part. This includes parts geometry, weight, fragility, flexibility and ease of handling. These are then stored as part modification in the parts database.

III. **Specify Liaison Changes:** data for the existing part liaisons is retrieved and updated to reflect the new requirements. This includes specification of the liaison type and constraints.

4.4.2 System Requirements Definition

As with system requirements for a new Reconfigurable Assembly System, The system requirements definition for system reconfiguration begins with a set of user requirements and ends with the formal specification of system requirements (see Figure 4-11).

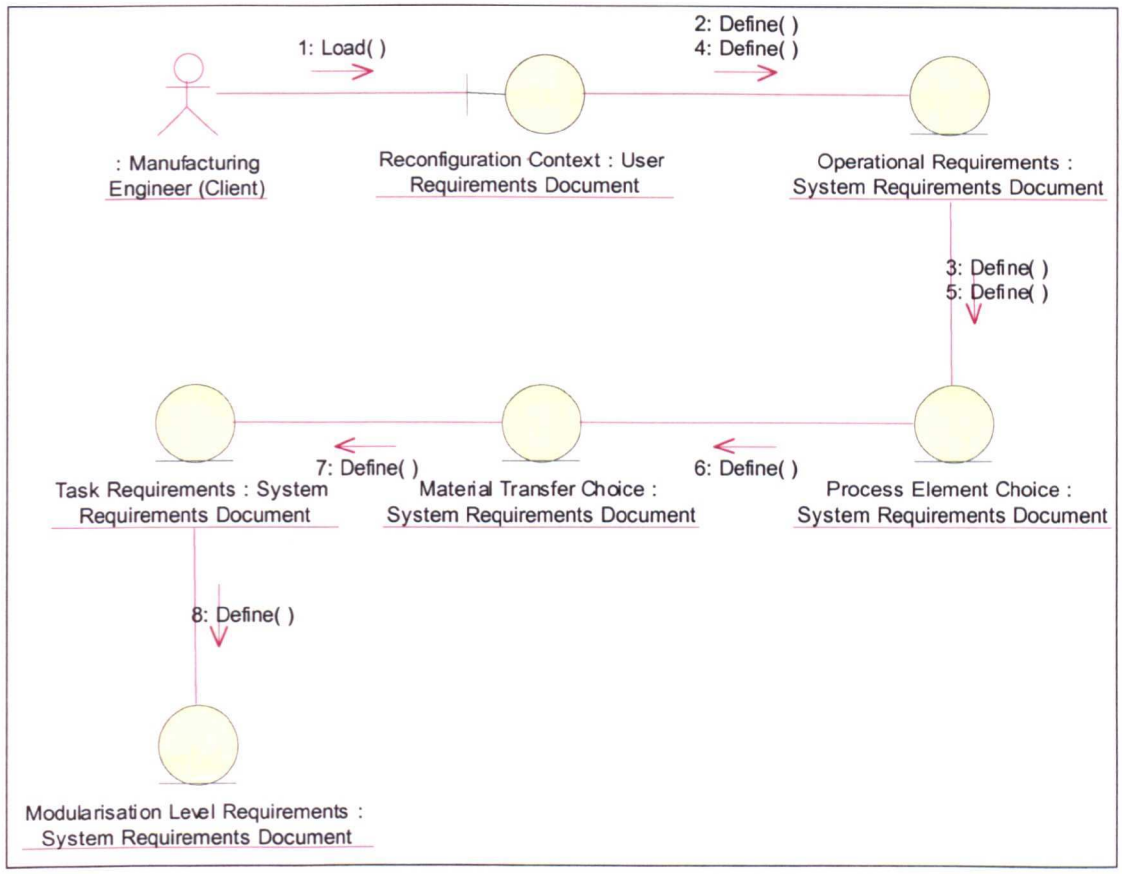


FIGURE 4-11: TASKS FOR SYSTEM REQUIREMENTS SPECIFICATION FOR SYSTEM RECONFIGURATION

Each task consists of a set of activities as described below:

I. **Load Reconfiguration Context:** This reveals the reason for the system reconfiguration that is whether it is for a new product or for a product modification. The execution of the following two tasks is determined by this fact.

II. **Define Operational Requirements** *(for new product)*: Part properties and part liaison characteristics are mapped to form the operational requirements for the product assembly.

III. **Define Operation Requirements** *(for existing product modification)*: Existing operational requirements are reviewed to find if they need to be updated due to the changes in parts and part liaisons.

IV. **Define Process Element Choice**[†] *(for new Product)*: process elements that belong to the operational requirements are chosen for the assembly task. A matching algorithm is executed that compares the required process elements for the new product to those already existing in the assembly system.

V. **Define Process Element Choice** *(for existing product modification)*: a matching algorithm is executed that compares process elements required to assemble the modified parts to those already in existence.

VI. **Material Transfer Choice:** the new parts are analysed to ensure that they are compatible to the material transfer system on the Reconfigurable Assembly System.

VII. **Task Requirements:** Assembly tasks are updated according to the changes imposed by the assembly requirements change.

VIII. **Modularisation level requirements:** process elements are analysed to ensure level of modularity available on the existing Reconfigurable Assembly System is suitable for the task changes that need to be incorporated.

Once the system requirements are defined they can then be used for the conceptual Reconfigurable Assembly System design. Until this stage they can all be referred for negotiation if the changes are deemed to be impractical by the systems integrator.

[†] The notion of process elements is introduced here – this is explained in section 6.

4.5 General Use Cases for Requirements Specification of Reconfigurable Assembly Systems

4.5.1 Requirements Analysis

Once the user requirements have been specified, they are sent to the system integrator for analysis. The analysis stage is mainly checking the feasibility of the user requirements against four main criteria. This is illustrated in Figure 4-12, for which the input is the user requirements document and the output is a set of requirements for negotiation and a set of approved requirements.

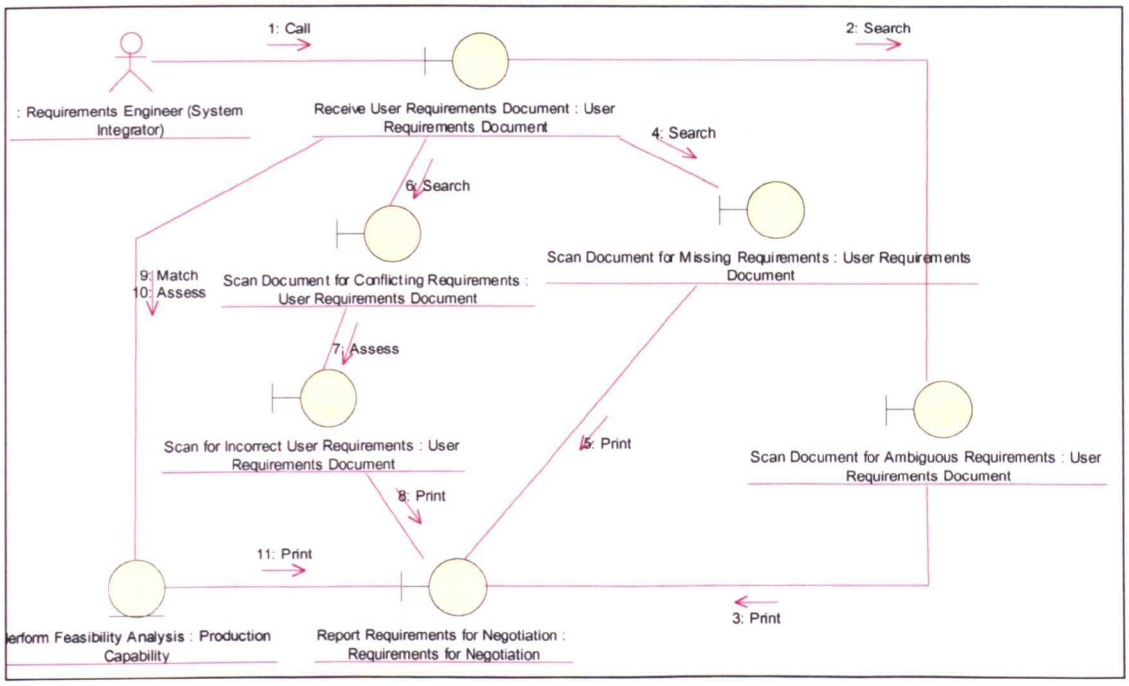


FIGURE 4-12: COLLABORATION DIAGRAM FOR REQUIREMENTS ANALYSIS

The sequence of activities for requirements analysis is as below:

I. **Receive user requirements:** the system integrator receives the user requirements document. This is analysed as per the analysis functions. There are two output types from this stage. The first is a list of requirements for negotiation, which is sent to the requirements negotiation process and the

other is the updated user requirements document, which is now ready for deriving system requirements.

II. **Scan document for missing requirements**: the user requirements document is searched for missing requirements. If essential information is omitted then the requirement is marked for negotiation, otherwise it is searched for ambiguity.

III. **Scan document for ambiguous requirements:** the user requirements document is searched for requirements that are unclear. Requirements that pass the test are sent to the next stage, whereas ambiguous requirements are marked and sent to the negotiation process.

IV. **Scan document for conflicting requirements:** requirements that have passed the previous two analysis stages are analysed for potential sources of conflict. This is matched to technical knowledge about how modular assembly systems are integrated and include information on compatibility and feasibility of design concepts. The problem of conflicting requirements is solved through requirements negotiation.

V. **Perform feasibility analysis:** the feasibility analysis stage is very system integrator specific. The aim is to consider the practical implications of the user requirements. Moreover this means matching the requirements to the system integrators abilities and assessing the project's feasibility. This is vital to ensure that the system integrator does not take unnecessarily risks by approving a project that they are incapable of satisfying the requirements for. Any points of concern are reported for negotiation.

VI. **Report requirements for negotiation:** the requirements to be negotiated are stated clearly with the reason for negotiation and a note stating any other relevant information that the system integrator deems necessary. Any resultant changes to the requirements are then fully traceable.

Any missing, ambiguous, conflicting or unfeasible requirements are sent back the system user for clarification through requirements negotiation.

4.5.2 Requirements Negotiation

Requirements Negotiation encompasses the resolution of outstanding requirements from the requirements analysis phase. Requirements that are reported for negotiation are processed according to the reason for negotiation. This is a consultation process between the system user and system integrator as illustrated in Figure 4-13.

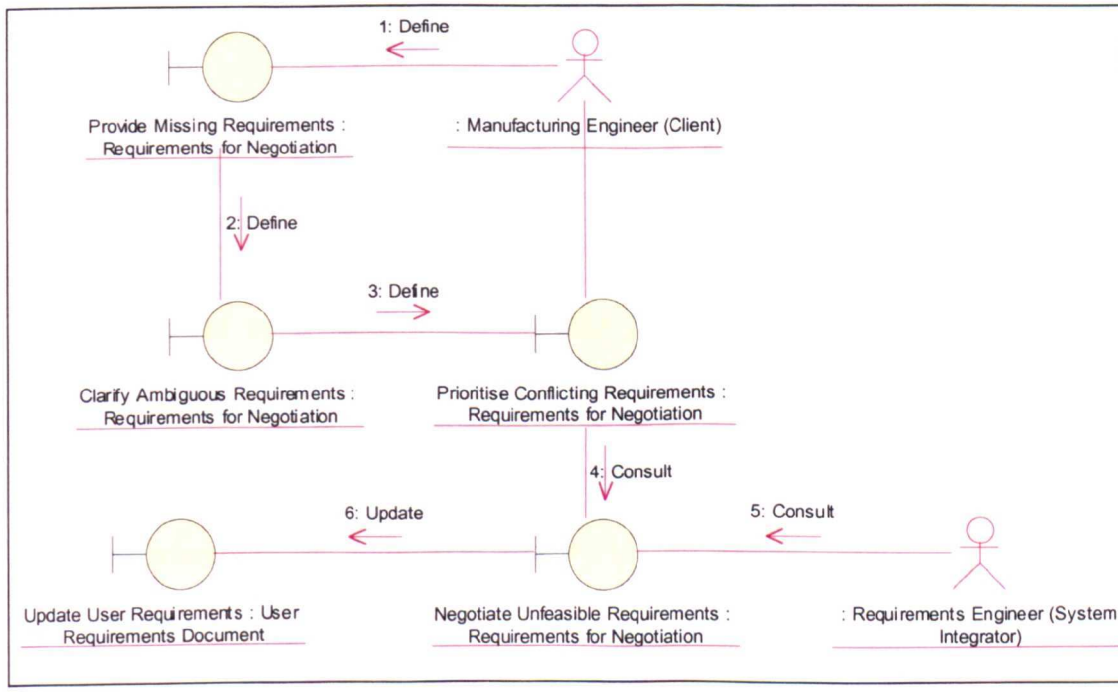


FIGURE 4-13: COLLABORATION DIAGRAM FOR REQUIREMENTS NEGOTIATION

The input for the requirements negotiation process is the list of requirements for negotiation. The system user performs the first three steps of providing missing requirements, clarifying ambiguous requirements and prioritising the conflicting requirements. The system integrator then enters the negotiation process and negotiates the unfeasible requirements with the system user. A description of each step is highlighted below:

I. **Provide missing requirements:** Requirements that were deemed as missing from the requirements analysis process are defined by the system user and confirmed by the system integrator.

II. **Clarify ambiguous requirements:** Ambiguous requirements are redefined and stated by the system user. The requirements are accepted if stated to a level that is acceptable to the system integrator.

III. **Prioritise conflicting requirements:** the reported conflicting requirements are prioritised and undergo negotiation until both parties are happy with the proposed solution.

IV. **Negotiate unfeasible requirements:** unfeasible requirements are either discarded or redefined to a feasible level. The system user and system integrator will do this manually whereby the system user will be considering the requirements of the assembly project and the system integrator will be examining its capability to deliver.

V. **Update user requirements:** once the requirements for negotiation have been resolved the user requirements document is updated so that all the requirements can be processed together to form system requirements and

ultimately develop an assembly system solution that satisfies the needs of the project.

Although it is recognised that the requirements negotiation process is a complex one and in practice determines the nature of assembly system development, only a simplified ideal of the process is presented here to maintain conciseness. The completion of this process results in a complete user requirements document, which is the basis for the system integrator to derive system requirements.

4.5.3 Requirements Verification

The requirements verification process considers the formalised system requirements and ensures that they match the system user's expectations of the assembly system. Figure 4-14 illustrates the requirements verification process.

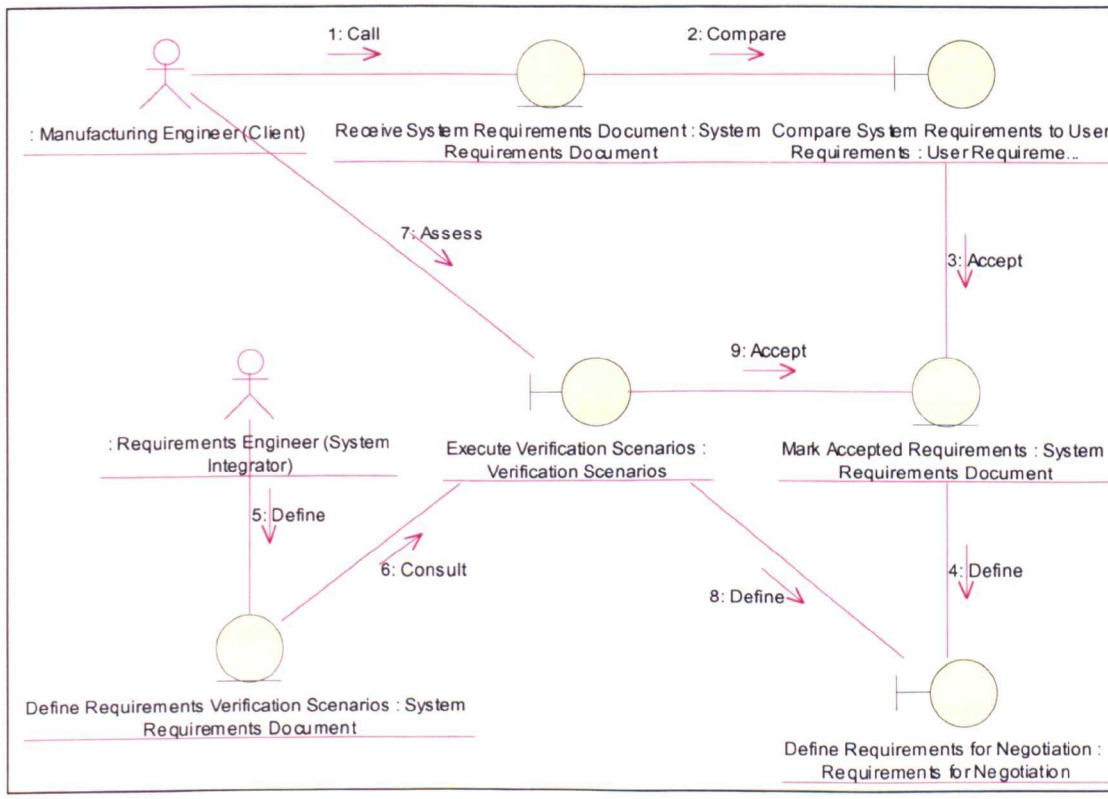


FIGURE 4-14: COLLABORATION DIAGRAM FOR REQUIREMENTS VERIFICATION

The completed system requirements document has to be verified by the system user. This can be used as a basis for system development only after it has been fully approved by the system user. This process is explained below:

I. **Receive system requirements document:** the system user receives the formalised system requirements document. This is either in electronic or printed form.

II. **Compare system requirements to user requirements:** the formalised system requirements are compared to the original user requirements to ensure that the system requirements are consistent with the original demands of the system user.

III. **Mark accepted requirements:** accepted requirements are marked as accepted by the system user. However, requirements that are not accepted are sent for requirements negotiation.

IV. **Define requirements for negotiation:** requirements that are unacceptable are marked and the reason for non-acceptance is stated in a message to be returned to the system integrator. These are then negotiated as per the requirements negotiation process.

V. **Define requirements verification scenarios:** the system integrator has the option of defining verification scenarios. These encompass actions that can occur within the scope of the system and those that are outside the scope as per the system integrators knowledge and experiences. They are discussed with the system user to define whether the system integrator's perception of the requirements for the project matches the system user's perception.

VI. **Execute requirements verification scenarios:** this is performed by the system user in conjunction with the system integrator. The implications of the verification scenarios are studied and requirements are redefined accordingly. The verification scenarios serve as a useful tool through which the two parties can communicate requirements to each other that are sometimes taken for granted and are sometimes not specified. Some requirements have to be re-negotiated as a result.

Once the requirements verification process has taken place and all requirements have been accepted, they can be passed on for conceptual design.

4.6 CHAPTER SUMMARY

A new Assembly System Requirements Specification Methodology has been developed including the following key stages: define user requirements; analyse user requirements; define system requirements; requirements negotiation; and requirements verification.

Using an object oriented approach user requirements have been described by classes relating to the product, its parts, liaisons between the parts and business constraints. The user requirements are then mapped to system requirements classes that refer to the assembly processes required to assemble the product within the business constraints. The system requirements data model supports the decision making for the specification of new Reconfigurable Assembly Systems and reconfigurations to existing Reconfigurable Assembly Systems.

The requirements analysis process created here involves identifying user requirements that are either missing ambiguous, conflicting or specified incorrectly. Any such requirements found are then sent for requirements negotiation between the system user and systems integrator. Requirements negotiation has been defined as a manual activity that involves the resolution of missing, conflicting, ambiguous or unacceptable requirements by the system user and systems integrator.

The requirements specification methodology includes requirements verification, which results in the acceptance or rejection of system requirements by the system user as the key activity. Any requirements that cannot be verified are negotiated by the system user and systems integrator.

5 KNOWLEDGE MODEL TO SUPPORT REQUIREMENTS SPECIFICATION OF RECONFIGURABLE ASSEMBLY SYSTEMS

5.1 INTRODUCTION

User requirements specification, user requirements analysis and system requirements specification provide the core decision making functions in the requirements specification for one of a kind Reconfigurable Assembly Systems. Hence the knowledge model created by the research describes task, domain and inference knowledge for these three aspects.

Task knowledge is specified in this thesis as part of the requirements specification methodology (see section 4), hence the domain knowledge that supports the tasks and the inferences (rules) that link the domain to the tasks are described here for user requirements specification, user requirements analysis, and system requirements specification of Reconfigurable Assembly Systems.

By definition domain knowledge is static data that is used or manipulated by a knowledge-based system (Schreiber, 1993). Hence the best way of representing this is through a database that allows easy search, retrieval and manipulation of items. A dedicated inference engine is then employed to perform the inference functions to retrieve, manipulate and save the domain knowledge.

During this research, domain knowledge has been compiled to form a domain knowledge schema. Items within the schema are described in this chapter. However, relationships between the individual items and a holistic view of the domain knowledge base is presented in Appendix B.

Inference rules are used to manipulate the domain knowledge and most of these are described in the main body of this chapter. However, the mapping inference rules that map user requirements to system requirements are presented in Appendix C, which includes a summary of the rules, followed by the rules written in JESS code.

Links between the domain, inference and task knowledge for each task are diagrammatically represented in Appendix D.

5.2 USER REQUIREMENTS SPECIFICATION

The goal of user requirements specification as stated in the requirements specification model and methodology is to specify the user requirements for Reconfigurable Assembly Systems. These are the characteristics of the product and conditions that the finished Reconfigurable Assembly System must fulfil to satisfy the system user.

5.2.1 Domain Knowledge for User Requirements Specification

Domain knowledge has been compiled and structured into database tables for user requirements specification of Reconfigurable Assembly Systems. Each table is populated by data from the system user. A holistic overview of the domains with the links between domains is illustrated in Appendix B. This is the information required by the systems integrator to provide a Reconfigurable Assembly System for the assembly of a product.

PROJECTS		
PROJ_ID	int(11)	<pk>
PROJ_NAME	varchar(100)	
PROJ_START	date	
PROJ_DUE	date	
PROJ_FINISH	date	
PROJ_BUDGET	decimal(10,2)	
PROJ_FAIL_RATE	int	
PROJ_SPACE_X	int	
PROJ_SPACE_Y	int	
PROJ_SPACE_Z	int	
PROJ_LIFESPAN	int	
PROJ_SHIFTS	int	
PROJ_OPERATORS	int	

ID: Unique identifier for fields in Projects domain
Project Name: Name assigned to the assembly system design project
Project Start Date: Proposed start date for the assembly system design project
Project Due Date: Proposed date of Assembly System delivery and installation
Project Finish Date: Actual date of fully working system installation
Budget: Amount of money the assembly system user is willing to pay for the system
Failure Rate: Acceptable level of defects while the system is running
Space X: Length of floorspace available in the system user facilities for the assembly system
Space Y: Width of floorspace available in the system user facilities for the assembly system
Space Z: Height of floorspace available in the system user facilities for the assembly system
Lifespan: Number of years the assembly system is expected to be in use
Shifts: Number of operator working shifts per day
Operators: Number of operators available to work on the assembly system

FIGURE 5-1: PROJECTS DOMAIN FOR USER REQUIREMENTS SPECIFICATION

The projects domain knowledge (Figure 5-1) contains data about individual projects, where each project refers to the design and development of a Reconfigurable Assembly System. These are the business requirements that are needed to perform the user requirements specification. Project timelines, budgetary requirements, spatial constraints, system lifespan and number of operators and shifts are stated in this domain. Projects are linked to Industrial Standards (Figure 5-2) through many-many links.

STANDARDS		
STAND_ID	int(11)	<pk>
STAND_NAME	varchar(150)	

ID: Unique identifier for fields in Standards domain
Standard Name: Name reference for an industrial standard

FIGURE 5-2: STANDARDS DOMAIN FOR USER REQUIREMENTS SPECIFICATION

Industrial standards that are relevant to the environment in which the Reconfigurable Assembly System will work are referred to in the user requirements. These can subsequently be found and included in the user requirements specification as any equipment that is used must conform to the standards laid out. Specification of industrial standards has been restricted to only naming the relevant standards. Inclusion of all aspects of an industrial standard would require a vast amount of investigation, which is not practical for the purpose of this research.

PRODUCTS		
PROD_ID	int(11)	<pk>
PROD_ST_ID	int(11)	<fk2>
DLV_METH_ID	int(11)	<fk1>
PROD_NAME	varchar(100)	
PROD_OVER_FUNC	varchar(200)	
PROD_VOLUME	int(10)	
PROD_PICNAME	varchar(200)	
PROD_BATCH	int	

ID: Unique identifier for fields in Products domain
Standard ID: Relational link to Industrial Standards domain
Delivery Method ID: Relational link to Delivery Method domain
Name: Name of product
Overall Function: Description of product use
Product Volume: Number of units to be assembled per annum
Product Picture Name: Reference to available pictures of product
Batch Size: Number of units to be packed in each batch

FIGURE 5-3: PRODUCTS DOMAIN FOR USER REQUIREMENTS SPECIFICATION

Each project includes the assembly of one or more products where each product has its own name, delivery method, batch size, overall function and volume (Figure 5-3). A diagram of the product may also be available.

PROD_STATUS		
PROD_ST_ID	int(11)	<pk>
PROD_ST_NAME	varchar(100)	

ID: Unique identifier for fields in Product Status domain
Status Name: Description of stage of development product is in

FIGURE 5-4: PRODUCT STATUS DOMAIN FOR USER REQUIREMENTS SPECIFICATION

The status of the product is specified because the product may only be designed at a conceptual level when the requirements for assembling it are specified. Consequently there is a higher probability of the product design changing and this must be considered at the system design stage. If there are physical models of the product or prototypes in existence, this is mentioned here so that the system integrator has a basis for initiating work. Moreover, the further down the development cycle the product is, the more concrete its design, allowing design decisions to be made with greater certainty as the nature of the parts is known. The ideal condition is if the product is already being assembled manually and a Reconfigurable Assembly System is needed to automate the process.

PARTS		
PAR_ID	int(11)	<pk>
MAT_ID	int	<fk1>
DLV_METH_ID	int(11)	<fk2>
PAR_NUMBER	varchar(200)	
PAR_NAME	varchar(100)	
PAR_ISCOMMON	char(1)	
PAR_WEIGHT	decimal(10,3)	
PAR_FRAGILITY	int	
PAR_SCRATCH	int	
PAR_VISIBILITY	int	
PAR_FLEXIBILITY	int	
PAR_HANDLING	int	
PAR_ORIENTATION	int	
PAR_BATCH	int	
PAR_PICNAME	varchar(200)	

ID: Unique identifier for fields in Parts domain
Material ID: Relational link to Material domain
Delivery Method ID: Relational link to Delivery Method domain
Number: Number of parts to be assembled into each product
Name: Name of part
Is Common: Description of part commonality
Weight: Approximate weight of part
Fragility: Estimation of propensity of part damage
Scratchability: Estimation of propensity of part scratching
Visibility: Estimation of ease of part visibility
Flexibility: Estimation of ability of part to change shape
Orientation: Estimation of ability for part to be oriented easily
Batch Size: number of parts to arrive at assembly system simultaneously
Picture Name: Reference to picture of part

FIGURE 5-5: PARTS DOMAIN FOR USER REQUIREMENTS SPECIFICATION

Each product is made up of many parts. It is the nature of the parts and the way in which they must be joined that determines the Reconfigurable Assembly System used. For this reason each part is described according to the criteria shown in Figure 5-5. Provisions have to be made for parts that are heavy, fragile, easy to scratch, flexible, difficult to handle or orientate. These qualities are built into the part descriptions so that the system integrator is aware of these factors.

LIAISONS		
LSN_ID	int	<pk>
ASM_PAR_ID	int	<fk>
LSN_PAR_ID	int	
LSN_TYPE	varchar(50)	

ID: Unique identifier for fields in Liaisons domain
ASM_Part ID: Relational link to part (A) involved in liaison
LSN_Part ID: Relational link to part (B) involved in liaison
Type: Description of liaison type

CONSTRAINTS		
CNS_LSN_ID	int	<pk,fk>
LSN_ID	int	<pk>
CNS_ACTION	varchar(20)	

ID: Unique identifier for fields in Constraints domain
Liaison ID: Relational link for liaison to place constraint on
Action: Description of type of constraint to be placed on liaison

FIGURE 5-6: PART LIAISON AND CONSTRAINTS DOMAINS FOR USER REQUIREMENTS SPECIFICATION

Liaisons describe how parts are joined together to form a product while constraints define restrictions that are imposed on liaisons (See Figure 5-6). For example if three parts (A, B and C) have to be joined together, these are described as two liaisons with the parts to be joined and the nature of the joints. If the joining of A and B has

to be performed before the joining of C to A and B, then this is a constraint imposed on the liaisons.

DELIVERY_METHOD		
DLV_METH_ID	int(11)	<pk>
DLV_METH_NAME	varchar(150)	

ID: Unique identifier for fields in Delivery Method domain
Name: Description of delivery method used

FIGURE 5-7: DELIVERY METHOD DOMAIN FOR USER REQUIREMENTS SPECIFICATION

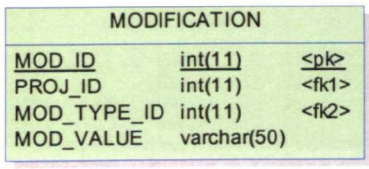
The information in Figure 5-7 is a subset of parts information as well as a subset of product as both parts and products are subject to similar delivery types. With parts, this refers to how they are delivered to the point of assembly. With products this refers to the state in which products leave the factory, that is, how they will be delivered to the customer, for example, bulk packed into boxes.

DIAGRAMS		
DGR_ID	int	<pk>
DGR_PARENT_ID	int	<pk,fk1,fk2>
DGR_PARENT	varchar(10)	<pk>
DGR_NAME	varchar(50)	
DGR_DESCRIPTION	text	
DGR_PATH	varchar(250)	

ID: Unique identifier for fields in Diagram domain
Parent ID: Relational link to any related parent diagrams
Parent: Name of related parent diagram
Name: Name of diagram
Description: Description of any notes to be assigned to diagram
Path: Reference to physical location of diagram on file

FIGURE 5-8: DIAGRAMS DOMAIN FOR USER REQUIREMENTS SPECIFICATION

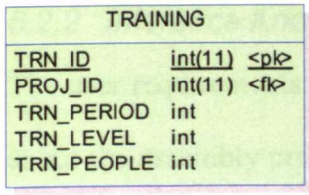
Diagrams are linked to both products and parts. Characteristics of diagrams are the same regardless of the context. This domain contains information on the location of the diagrams (Figure 5-8) so that they can subsequently be located and displayed. The diagram description allows the system user to write a note related to the diagram to make it clearer for the systems integrator to understand.



ID: Unique identifier for fields in Modification domain
Project ID: Relational link to Project with modified requirements
Type: Type of modification needed
Value: Description of modification needed

FIGURE 5-9: FUTURE MODIFICATION DOMAIN FOR USER REQUIREMENTS SPECIFICATION

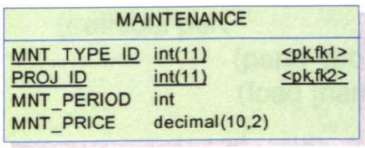
Future modification data refers to expected system modifications that will have to be accommodated for future change. The modification type refers to whether the change is going to be at a system, product or process level. The value allows the entry of data specific to the project so that a record exists of the future reconfigurations that will have to be accommodated in the Reconfigurable Assembly System (Figure 5-9).



ID: Unique identifier for fields in Training domain
Project ID: Relational Link to Project for which training is provided
Period: Length of time training will take place
Level: Level of expertise that personnel will be trained to
People: Number of people that will be trained

FIGURE 5-10: TRAINING DOMAIN FOR USER REQUIREMENTS SPECIFICATION

Training reflects the different types of training packages that are available. The amount of time, level of training and the number of people is included for each instance (see Figure 5-10). The level of training refers to the content of the training given, for example, whether operators are merely trained to perform the day to day operations and maintenance, minor repairs or given full knowledge of how the Reconfigurable Assembly System is built and programmed so that they can carry out their own system reconfigurations in the future.



ID: Unique identifier for fields in Maintenance domain
Project ID: Relational link to Project for which maintenance is provided for
Period: Length of maintenance contract
Price: amount of remuneration for maintenance contract

FIGURE 5-11: MAINTENANCE DOMAIN FOR USER REQUIREMENTS SPECIFICATION

Maintenance includes the period of time during which the Reconfigurable Assembly System will be serviced by the system integrator. This refers to the combination of parts and labour cost cover for maintenance (Figure 5-11). In some cases maintenance will also include exclusive rights to carry out system reconfiguration or supply of modules for system reconfiguration.

5.2.2 *Inference Knowledge for User Requirements Specification*

The user requirements specification is mainly concerned with providing information about the assembly project; hence the inferences here refer to data entry and storage activities.

5.2.2.1 Enter Data

The enter data inference prompts the user to enter data within a field that has been recalled from the domain structure. This is implemented by the "deffact ()" function in the JESS expert system shell, which allows the user to define facts.

The process is normally preceded by a pointer to domain and results in the saving of data as part of the deffact() rule. The data can then be called and manipulated as required. An example is shown in Figure 5-12:

```
(deffacts part
                (parts_props (name ?)
                (load [name])
                )
                (delivery ?) (weight ?) (fragility ?)(scratch ?)(visible ?)
                     (flexible?) (Handle ?)(orient ?)(batch_si ?)(pic ?)
)
```

FIGURE 5-12: JESS CODE FOR ENTER DOMAIN INFERENCE

5.2.2.2 Pointer to Domain

The pointer to domain provides the link between the task and the domain knowledge associated with that task. This is implemented in JESS through the defrelation () function. An example of this is illustrated in Figure 5-13.

```
(defrelation part_supply (?))
(defrelation feeding_rate (?))
(defrelation partlink (?))
(defrelation movement (?))
(defrelation machines (?))
(defrelation method (?))
(defrelation transfer_system (?))
(defrelation reconfiguration_type (?))
```

FIGURE 5-13: JESS CODE FOR POINTER TO DOMAIN INFERENCE

Each defrelation command specifies a link to a field (slot) within the tables in the domain structure. Execution of the task automatically initialises the pointer to the domains. The domain that is required for the execution of the task is returned and the relevant data from the domain is extracted.

5.2.3 Integration of User Requirements Specification Knowledge

The above knowledge is needed to perform the user requirements specification use case. Each domain is linked to the task level knowledge through inference rules. The main knowledge function being carried out here is that of data entry and storage. A complete mapping illustrating the relationships between tasks, domain and inference rules for each use case is presented in Appendix D. The requirements specified through the user requirements specification are analysed for irregularities in the User Requirements Analysis.

5.3 USER REQUIREMENTS ANALYSIS

The goal of the user requirements analysis is to ensure that requirements passed on for system specification are defined to the necessary level of detail. This is to ensure that system specification time and therefore cost is spent exclusively on satisfying concrete requirements and not on decisions that have to be revised later due to avoidable errors in the requirements specification.

5.3.1 Domain Knowledge for User Requirements Analysis

The domain knowledge needed in the user requirements analysis is that which is already specified in 5.2.1. In this use case the data that has been entered in the user requirements specification is analysed and any irregularities are reported for requirements negotiation. These are reported through a domain structure as illustrated in Figure 5-14.

REQS_NEGOTIATION		
NUM_R_NEG_ID	int(11)	<pk>
SR_ID	int(11)	<fk1>
PROJ_ID	int(11)	<fk2>
REQ_NAME	varchar(15)	
REASON	longtext	
MODIFIED	int zerofile	
DATE_MODIFIED	date	
MODIFIED_BY	varchar(10)	
NOTES	longtext	

ID: Unique identifier for fields in Requirements_Negotiation domain
System Requirements ID: Relational link to system requirement that call for negotiation is made from
Project ID: Relational link to Project that the call for negotiation is concerned with
Requirement Name: Name of the requirement that needs to be negotiated
Reason: Statement of the reason for the negotiation
Modified: Description of the requirement after negotiation
Date Modified: Date that the requirement was negotiated and subsequently modified
Modified By: Name of person that performed the requirement modification
Notes: Any additional information concerning requirements negotiation for this item

FIGURE 5-14: REQUIREMENTS NEGOTIATION FOR SYSTEM REQUIREMENTS SPECIFICATION

The requirements negotiation structure captures the reasons for requirements not being approved by either the system user or systems integrator, and any resulting modifications are logged for future reference (Figure 5-14). This includes references to the project and system requirements that are affected by the negotiation as well as the date of modification.

5.3.2 Inference Knowledge for User Requirements Analysis

Inference knowledge for this use case is concerned with scanning data and confirming that it adheres to expectations in that it exists, it is of the correct type and that it does not conflict with other data.

5.3.2.1 Scan – Missing Field

The scan – missing field inference searches through each of the data contained in the domains specified by the pointer to domain and reports on any essential fields that do not contain any data. This is performed by the slotboundp() function in JESS as demonstrated in Figure 5-15.

```
(slot-boundp (?))
(IF     (slotboundp = FALSE)
=>
(printout t  (slot) " is missing")
)
```

FIGURE 5-15: JESS CODE FOR SCAN MISSING FIELD INFERENCE

Input to the scan – missing field inference is the set of domains that have been populated in the user requirements specification use case. Fields that do not have any entries are reported for requirements negotiation. The enter data inference is fired for these fields.

5.3.2.2 Scan – Incorrect Data Type

The scan – incorrect data type inference searches through each of the data contained in the domains specified by the pointer to domain and reports on any requirements that have been specified with the incorrect data type. For example if text is entered into a field instead of a number. This can be checked by several functions the JESS expert system shell as shown in Figure 5-16

```
(IF     (numberp <budget> = FALSE)
=>
(printout t "incorrect data type – please enter a number into the budget field")
(modify slot (budget))
)
```

FIGURE 5-16: EXAMPLE OF JESS RULE FOR SCAN INCORRECT DATA TYPE INFERENCE

The example shown above is for the entry of the budget field. If a character other than a number is entered into the field, this is detected and the user is asked to enter a valid number for the budget.

Input to the scan – incorrect data type inference is the set of domains that have been populated in the user requirements specification use case. Fields that contain incorrect data entries are reported for requirements negotiation. The enter data inference is initialised for these fields and user is prompted to supply the missing data.

5.3.2.3 Scan – Sources of Conflict

Sources of conflict are defined by pointers to domains that have been identified as potential sources of conflict. It is worth noting that conflicts can occur between data in different domains as well as in the same domain. Data values in these fields are extracted and tested for conflict as per the example in Figure 5-17.

```
(IF     (slot (cost)) > (slot (budget))
=>
(printout t "Cost is above Budget. The requirement is referred for
prioritisation")
(assert priority ())
)
```

FIGURE 5-17: EXAMPLE OF JESS RULE FOR SCAN SOURCES OF CONFLICT INFERENCE

The inputs to this inference are the data items from the domains that are identified as potential sources of conflict. In the example above the case of the cost not meeting the budget is taken. When the inference engine finds this case, it sends out a message acknowledging the find and calls the assign priority inference.

5.3.2.4 Compare

The compare inference takes two or more like values specified by the pointer to domain and compares them. This encompasses matching data fields or evaluating whether the value entered in one field is greater than, equal to, or less than the value of another field. For qualitative data only equal to or not equal to relationships are compared. This function is implemented through the following command (Figure 5-18) in JESS.

```
(Compare (mach_z)(floor_spc_z))
```

FIGURE 5-18: JESS CODE FOR COMPARE INFERENCE

In the example shown in Figure 5-18 a comparison is made between the height of the equipment and the space available in the room to accommodate that height. The same principle is applied to checking any user requirements that map to system requirements on a one-to-one basis.

Input to the compare inference is two fields that are specified by the pointer to domain under the relevant task. Output from the inference is a set of results that report the status of the comparison made.

5.3.2.5 Assign Priority

The input to this inference is a set of conflicting requirements that are reported for prioritisation. The user declares the level of preference that can be given to the conflicting requirements. Output from the inference is that the requirements can be gauged and preferences are declared for conflict resolution. An example is shown in

Figure 5-19.

```
(assert slot (priority <int>)
```

FIGURE 5-19: JESS CODE FOR ASSIGN PRIORITY INFERENCE

A priority is assigned to the conflicting requirements in the form of an integer. The lowest value integer takes precedence when requirements are negotiated.

5.3.3 Integration of User Requirements Analysis Knowledge

In summary these five inferences complete the user requirements analysis. The relationships between the domains, tasks and inferences in user requirements analysis is illustrated in Table 5-1.

Task	Domain Needed	Inferences	
Scan Document for Missing Requirements	All	Scan – missing field	Pointer to domain
Scan Document for Ambiguous Requirements	All	Scan – incorrect data type	Pointer to domain
Scan Document for Conflicting Requirements	All	Scan – sources of conflict	Pointer to domain
Receive User Requirements	All	Load	Pointer to domain
Perform Feasibility Analysis	Own project requirements	Compare project credentials to own criteria for project selection	Pointer to domain
Report Requirements for Negotiation	Requirements for Negotiation	Define	Flag and copy

TABLE 5-1 : KNOWLEDGE MODEL SUMMARY FOR USER REQUIREMENTS ANALYSIS

The result of the user requirements analysis is a set of requirements that are sent for negotiation together with the requirements that are ready for system requirements specification. Once all the tasks in this use case have been completed, requirements negotiation and system requirements specification can begin. Requirements negotiation is not included in the knowledge model as it is generally a manual process and is too user specific to account for all the factors in a knowledge model.

5.4 SYSTEM REQUIREMENTS SPECIFICATION

The goal of system requirements specification is to specify the system properties to a suitable level of abstraction so that system designers have clear boundaries to work within whilst maintaining their freedom of expression to deliver novel and innovative solutions. The domain and inference knowledge that underpins this is presented in this section.

5.4.1 Domain Knowledge for System Requirements Specification

The domain knowledge for system requirements specification helps the system integrator to define the boundaries of the system by providing a pool of data. For system requirements specification the pool of data has two components:

- Static data that is used to describe solutions (noted in yellow)
- Semi-static data that serves as a storage medium for the current project (noted in green).

The distinction between the two domain types is that the user can edit the semi-static domain knowledge, whereas only a system administrator can edit the static domain. In most cases static data is extracted from the domain base to populate the tables of semi-static data.

SYSTEM_OVERVIEW		
SR_ID	int(11)	<pk>
SR_ARCHID	int(11)	<fk1>
MAT_TRAN_ID	int(11)	<fk2>
PROJ_ID	int(11)	<fk3>
SR_SYS_CONCEPT	varchar(100)	
SR_VOLUME	int(11)	
SR_SPEED	int(11)	
SR_ACCURACY	int(11)	
SR_OVERSIZE_X	int(11)	
SR_OVERSIZE_Y	int(11)	
SR_OVERSIZE_Z	int(11)	
SR_COST	decimal(10,2)	

ID: Unique identifier for fields in System_Overview domain
SR_ARCHID: Relational link to control architecture domain
MAT_TRAN_ID: Relational link to material transfer system domain
PROJ_ID: Relational link to Project domain
System Concept: description of the system concept
Volume: Number of good parts to be produced by the assembly system in a year
Speed: Speed at which the assembly system must produce good parts at
Accuracy: Percentage of good products as a proportion of total products to be assembled
Overall Size X: Maximum length of assembly system
Overall Size Y: Maximum width of assembly system
Overall Size Z: Maximum height of assembly system
Cost: Estimated cost of assembly system design and implementation project

FIGURE 5-20: SYSTEM OVERVIEW DOMAIN FOR SYSTEM REQUIREMENTS SPECIFICATION

The system overview domain (see Figure 5-20) is a store of information that can forms a useful project overview for the systems integrator. As the name suggests, it presents an overview of what the system contains. Moreover it outlines the global constraints that are placed on the assembly the system being developed. Links to other domains are also stated to make information traceable.

TASKS		
NUM_TASKID	int	<pk>
NUM_FEEDID	int	<fk1>
NUM_SYSREQID	int(11)	<fk2>
TXT_NAME	varchar(50)	

ID: Unique identifier for fields in Tasks domain
FEEDID: Relational link to feeding system domain
SYSREQID: Relational link to System Requirements domain
Name: Name assigned to assembly task

FIGURE 5-21: ASSEMBLY TASKS FOR SYSTEM REQUIREMENTS SPECIFICATION

An assembly system performs a set of tasks where each task involves the addition of a part to the product. The task description includes the presentation of the part (feeding), the part descriptions, the relationship between the parts and the operations that are needed to assemble the parts.

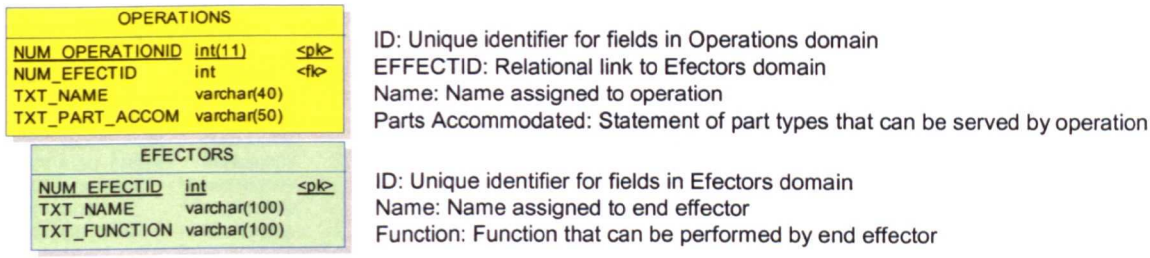


FIGURE 5-22: ASSEMBLY OPERATIONS AND EFFECTORS FOR SYSTEM REQUIREMENTS SPECIFICATION

The description of assembly operations includes the properties of the manipulators that are required to perform the operation together with a description of the type of materials that can be accommodated (see Figure 5-22). This is to allow the combination of different assembly modules, for example, insertion operations can be performed with several different grippers including two finger, three finger and electromagnetic and so on.

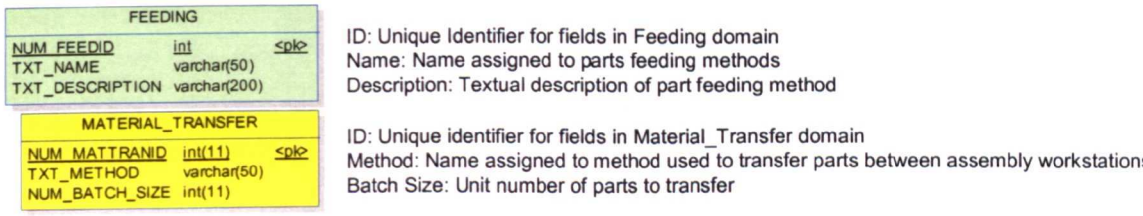


FIGURE 5-23: FEEDING AND MATERIAL TRANSFER FOR SYSTEM REQUIREMENTS SPECIFICATION

Methods of feeding parts are described with a name and description of the feeding method as shown in Figure 5-23 while the material transfer table includes a description of the method of transfer and batch size. The distinction between the two is that feeding encompasses the supply of parts to the assembly machine whereas material transfer describes how parts and products are transferred between workstations on the machine.

CONTROL_ARCH		
NUM_ARCHID	int(11)	<pk>
TXT_NAME	varchar(20)	
TXT_GLOB_BRAND	varchar(60)	
TXT_GLOB_NUM	varchar(60)	
TXT_LOC_BRAND	varchar(60)	
TXT_LOC_NUM	varchar(60)	
TXT_BUS	varchar(50)	
TXT_OP_SYS	varchar(30)	

ID: Unique identifier for fields in Control_Arch domain
Name: Name assigned to control architecture
Global Brand: Brand name of global control unit used
Global Num: Number of global control units used
Local Brand: Brand name of local control units used
Local Number: Number of local control units required
Bus: Type of bus used to integrate control units
Operating System: Type of operating system used by control architecture

FIGURE 5-24: CONTROL ARCHITECTURE FOR SYSTEM REQUIREMENTS SPECIFICATION

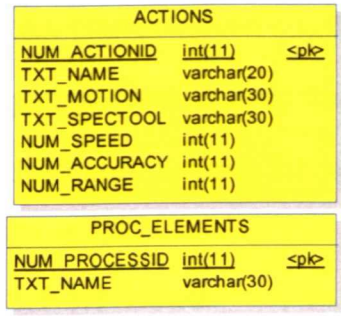
Domain knowledge regarding the control architecture aims to describe the structure of machine control most suitable for the assembly project. This includes descriptions of the global controller that synchronises the various parts of the machine, the local controller that controls individual workstations, the computer environment that they work in and the interfaces between the various control aspects.

OPERATIONAL_REQS		
NUM_OPERREQID	int(11)	<pk>
NUM_ACTIONID	int(11)	<fk1>
NUM_PARTID	int(11)	<fk3>
NUM_OPERATIONID	int(11)	<fk2>
TXT_FUNC_TEST	varchar(250)	
NUM_WORKENV_X	int(11)	
NUM_WORKENV_Y	int(11)	
NUM_WORKENV_Z	int(11)	

ID: Unique identifier for fields in Operational_Reqs domain
ACTIONID: Relational link to assembly Actions domain
PARTID: Relational link to Parts domain
OPERATIONID: Relational link to Operations domain
Functional Test: Description of type of functional test required to test operation success
Work Envelope X: working length needed for assembly operation
Work Envelope Y: working width needed for assembly operation
Work Envelope Z: working height needed for assembly operation

FIGURE 5-25: OPERATIONAL REQUIREMENTS FOR SYSTEM REQUIREMENTS SPECIFICATION

The Operational Requirements domain (Figure 5-25) records information regarding how parts are assembled on the line including the assembly operations and actions that are required. The working envelope and functional test aspects are also covered in this domain.

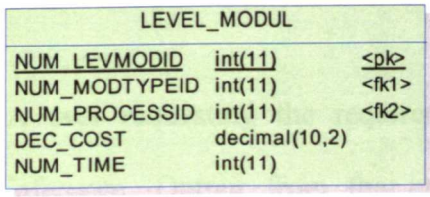


ID: Unique identifier for fields in Actions domain
Name: Name assigned to assembly action
Motion: Type of movement performed by assembly action
Special Tool: Name of any special tools required to perform assembly action
Speed: Cycle time within which assembly action can be performed
Accuracy: Tolerance within which assembly action can be performed
Range: Range of movement performed by assembly action

ID: Unique identifier for fields in Proc_Elements domain
Name: Name assigned to process element

FIGURE 5-26: ASSEMBLY ACTIONS AND PROCESS ELEMENTS FOR SYSTEM REQUIREMENTS SPECIFICATION

Assembly actions and process elements describe the movements and their parameters (Figure 5-26) that are needed to assemble the product. These are stores of static data that are selected to subsequently form part of the operational requirements.



ID: Unique identifier for fields in Level_Modul domain
MODTYPEID: Relational link to module type domain
PROCESSID: Relational link to Process domain
Cost: Estimated cost of providing level of modularition
Time: Estimated setup time for integrating module

FIGURE 5-27: LEVEL OF MODULARISATION FOR SYSTEM REQUIREMENTS SPECIFICATION

Level of Modularisation refers to the interchangeability of modules that can be accommodated by a Reconfigurable Assembly System. The time and cost incurred by the reconfiguration and the module type and relevant process elements form the knowledge in this domain (Figure 5-27).

5.4.2 Inference Knowledge for System Requirements Specification

Domain knowledge is supported by inferences that provide the link between the user requirements specification and the system requirements specification.

User requirements that are gathered, analysed and negotiated are parsed to derive system requirements in the system requirements specification. The parsing functions are performed by the inferences described in this section, which take user

requirements, compare them with the domain knowledge contained in 5.4.1 and generate new domain knowledge, which forms the system requirements document that forms the basis for conceptual system design.

5.4.2.1 Load

Requirements are loaded from a file. This includes domain data specific to an application or project. The load inference is enacted by the load function in JESS as shown in Figure 5-28.

```
(Load [filename])
```

FIGURE 5-28: JESS CODE FOR LOAD INFERENCE

A user requesting the requirements documents for a project initiates the load inference. Output from the inference is access to the requirements document requested by the user.

5.4.2.2 Map

If:then rules are used to assert system requirements in response to the user requirements. This forms the greatest functionality relevant to the requirements specification for Reconfigurable Assembly Systems and has been implemented for the mapping of user requirements to choose a system concept.

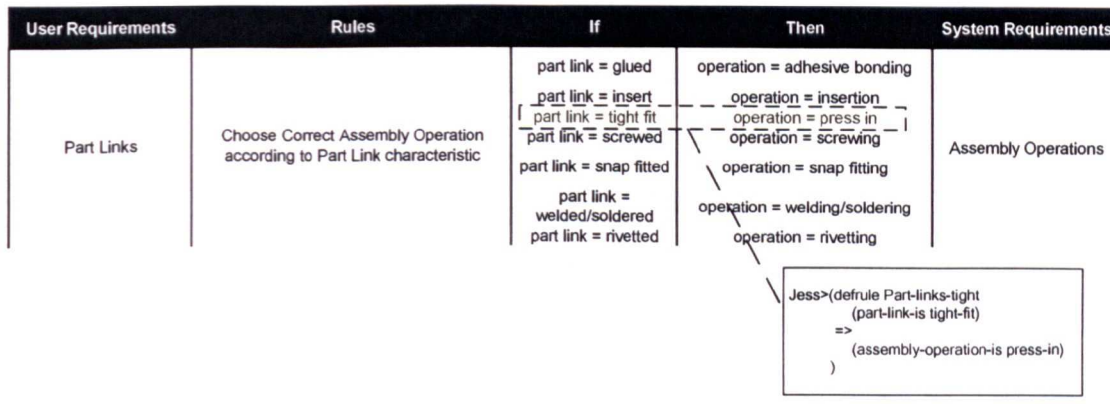


FIGURE 5-29: MAP INFERENCE RULES FOR CHOOSING OPERATIONS

The choose operation mapping inference takes the part links that are described in the user requirements specification and returns the operations that can achieve these links in the system requirements. Figure 5-29 illustrates the generic textual description of the rule, which is to choose the assembly operation that corresponds to the nature of the link between two parts. The "if" and "then" columns summarise the inference and show that if the part link is tight fit, then the assembly operation that would be required would be press in and so forth. A JESS rule called Part-links-tight has been defined to execute the inference function, where the variable part-link is loaded and checked to see if it is called tight-fit. If it is called tight-fit, then a variable called assembly-operation is loaded and the string press-in is inserted to define this variable.

The mapping inference for choosing the feeding analyses the part to be fed in terms of how it is supplied, and its properties and then determines the most appropriate feeding method based on this information.

User Requirements	Rules	If	Then	System Requirements
Parts (All Data)	Feeding - when part supply is pre-defined	part is manufactured on-line	feeding = on-line manufacturing	Part Feeding
		part is supplied in pallets	feeding = pallet feeding	
		part is supplied in tape	feeding = tape feeding	
		part is supplied in bulk bags/boxes	feeding = bowl feeding	
		part is liquid	feeding = Liquid Dispensing	
		part is supplied on film	feeding = film feeding	
		part is supplied in foil	feeding = foil feeding	
		part is supplied in magazines	feeding = magazine	
		part is supplied in trays	feeding = tray feeding	
	Feeding - when part supply is not pre-defined	assembly can be integrated into manufacturing of the	feeding = Pallet feeding OR on-line manufacturing	
		part is manufactured on-line	feeding = on-line manufacturing	
		part is small	feeding = tape feeding	
		part is easy to scratch	feeding =! bowl feeding	
		part is easy to orientate automatically	feeding = bowl feeding	
		part is fragile	feeding = tray feeding	
		part is easy to scratch AND scratchability cannot be limited	feeding = tray feeding OR magazine feeding	

```
Jess>(defrule feeding-not-defined-scratch
        (part-is easy-to-scratch)
      =>
        (feeding-is !bowl-feeding)
      )
```

FIGURE 5-30: EXTRACT OF MAPPING INFERENCE RULES FOR CHOOSING FEEDING

Figure 5-30 shows an extract from the feeding inferences summary and the JESS inference rule for executing one of the rules. Part feeding is largely defined by the method of part delivery so if the part delivery has already been defined then the part feeding method can be selected accordingly. For example if the part is going to be supplied in trays, then tray feeding will be used. However, if the part feeding has not been defined, or if there is a possibility to negotiate the supply method, then part properties need to be taken into consideration and these can be used to select the most appropriate part feeding method. The example highlighted in Figure 5-30 illustrates that bowl feeding should not be used if the part is easy to scratch. Implementation of the rule in JESS is illustrated where the inference engine loads the part domain and checks the scratchability level entered. If the part is easy to scratch then bowl feeding of the part will be prevented.

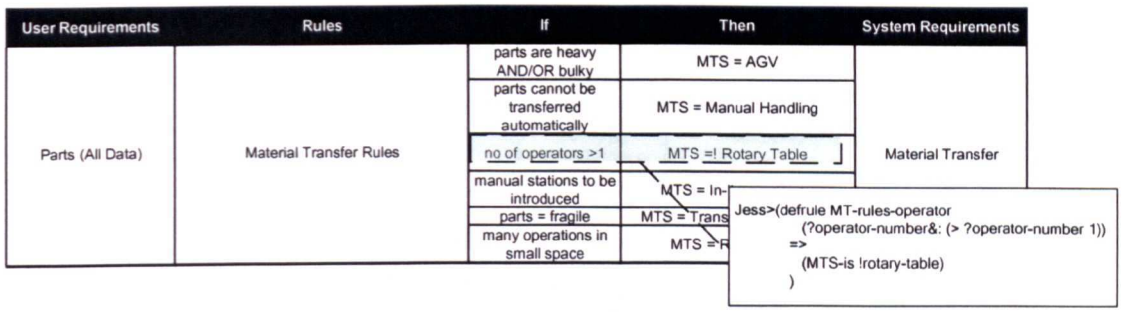


FIGURE 5-31: EXTRACT FROM MAPPING INFERENCE RULES FOR CHOOSING MATERIAL TRANSFER SYSTEM

The choose transfer mapping inference analyses the properties of the parts to be transferred in the assembly machine and determines the most suitable material transfer method. Figure 5-31 illustrates that data under the parts domain is loaded and this is then mapped to the material transfer domain. If the number of operators is greater than one then a rotary table is not suitable as the method of material transfer. The JESS rule MT-rules-operator is used to execute this inference where the inference engine searches for a number within the number of operators field in the domain. Once this has been found the inference checks the number to see if it is greater than one. If the number of operators needed is greater than one then the material transfer domain will not allow a rotary table to be used.

User Requirements	Rules	If	Then	System Requirements
Legacy Systems	System Concept should reflect companies' overall strategy	High Turnover of Products Product Parts Changing Processes Changing	Do not specify fixed assembly Product Level Modularity Process Level Modularty	Level of Modularisation
Future Modifications	LoM = None of no future modifications LoM = Product if minor changes are made to products LoM = Process if product changes result in process changes LoM = System if changes in products are significant and/or volume fluctuations are large	FM = None Minor changes to products Changes in Processes Significant product/volume changes	No Modularity Required Product Level Modularity Process Level Modularity System Level Modularity	
System Lifespan	Longer Lifespan = Higher Level of Modularisation			
Production Volume	System Concept should accommodate the volume and variety of parts being assembled	High Turnover of Products Product Parts Changing Processes Changing	Do not specify fixed assembly Product Level Modularity Process Level Modularty	
Future Modifications	Not Fixed Automation if there are Future Modifications.	FM = yes	Reconfigurable Assembly Systen	

```
Jess>(defrule LoM-lifespan5
        (lifespan-is >5)
      =>
        (LoM-is !product)
        (LOM-is !none)
      )

Jess>(defrule LoM-lifespan10
        (lifespan-is >10)
      =>
        (LoM-is !process)
      )
```

FIGURE 5-32: EXTRACT OF MAPPING INFERENCE RULES FOR CHOOSING LEVEL OF RECONFIGURATION REQUIRED

The choose level of reconfiguration mapping rule looks at the future demands that may be placed on the system and suggests the most appropriate level of reconfiguration to adopt to satisfy these requirements. Figure 5-32 presents an extract from the summary of mapping rules and a demonstration of one of the rules implemented in JESS. If the system lifespan is longer then more modularity needs to be built into the Reconfigurable Assembly System. This statement has been decomposed to suggest that product level or no modularity should not be used when the system lifespan is greater than five years and that process level modularity should not be used when the system lifespan is greater than ten years. The JESS rule LoM-lifespan5 scans the system lifespan entry in the domain knowledge and if this is greater than 5 years, then it prevents the level of modularisation from being defined as product or none levels. Furthermore, the LoM-lifespan10 rule checks the system lifespan entry in the domain knowledge and if this number is greater than ten, prevents level of modularisation from being defined as process level.

A closer look at the four instances of mapping inference rules reveals that input to the inference is from a set of domains stated by pointers to domains in the task description. Output from the inferences is a set of data that has been induced through reading data in those domains. This is then saved to the relevant domains in the domain tables. A summary of the mapping rules and JESS code for executing the mapping rules is presented in Appendix C.

The great advantage of the rule-based mapping is that rules can be expanded or modified easily to accommodate new methods as and when they become available. Each rule is independent so modification of one rule has a minimal knock on effect.

5.4.2.3 Confirm

The confirm inference marks data entries that have been confirmed and accepted by a stakeholder. This inference needs a data entry to confirm, which is provided by the pointer to domain. Once an entry has been confirmed it is then saved (Figure 5-33).

```
(Modify (confirmed Y))

(Save [requirements filename])
```

FIGURE 5-33: JESS CODE FOR CONFIRM INFERENCE

The confirmed slot in the instance of each requirement is modified to say yes and the requirements are saved into a filename as per Figure 5-33. Once all the requirements have been confirmed and saved they can be sent for verification.

5.4.3 Integration of System Requirements Specification Knowledge

The system requirements specification performs the vast majority of knowledge-enriched functions within the knowledge model. These are summarised in Table 5-2, which presents the domain and inference mappings according to the task classification as outlined in the requirements specification methodology (see section 4).

Task	Domain Needed	Inferences	
Consult User Requirements Document	All Domains	Load document	
Identify Redundant Requirements	All Domains	Manual process	
Map User Requirements to System Requirements	All Domains	Map - as per requirements mapping guide	Define links in mapping guide – if:then rules
Formalise System Requirements	All Domains	Confirm and save	Pointer to domain

TABLE 5-2: KNOWLEDGE MODEL SUMMARY FOR SYSTEM REQUIREMENTS SPECIFICATION

A complete listing of the relationships between tasks, inferences and domains is illustrated in Appendix D.

5.5 CHAPTER SUMMARY

A new knowledge model has been created whereby domain and inference knowledge has been structured to support the execution of tasks in user requirements specification, requirements analysis and system requirements specification.

"Pointer to domain" and "save data" inference rules have been proposed and a domain knowledge base has been developed to support user requirements specification. The domain knowledge stores user requirements under the following categories: project requirements; products; parts; and liaison characteristics.

Inference rules have been created that scan through domain knowledge to find missing requirements, incorrect data types and other conflicting requirements. Any requirements that fit these criteria are sent for additional negotiation by the system user and systems integrator.

User requirements which are stored in the domain knowledge are mapped to system requirements through mapping inference rules. New domain knowledge is created that describes the project from a systems point of view, including project requirements and assembly task requirements. Hence domain and inference rules have been specified so that they can be easily updated as new assembly processes are developed or as new knowledge is discovered.

6 ASSEMBLY SYSTEM CAPABILITY MODEL FOR REQUIREMENTS SPECIFICATION OF RECONFIGURABLE ASSEMBLY SYSTEMS

6.1 INTRODUCTION

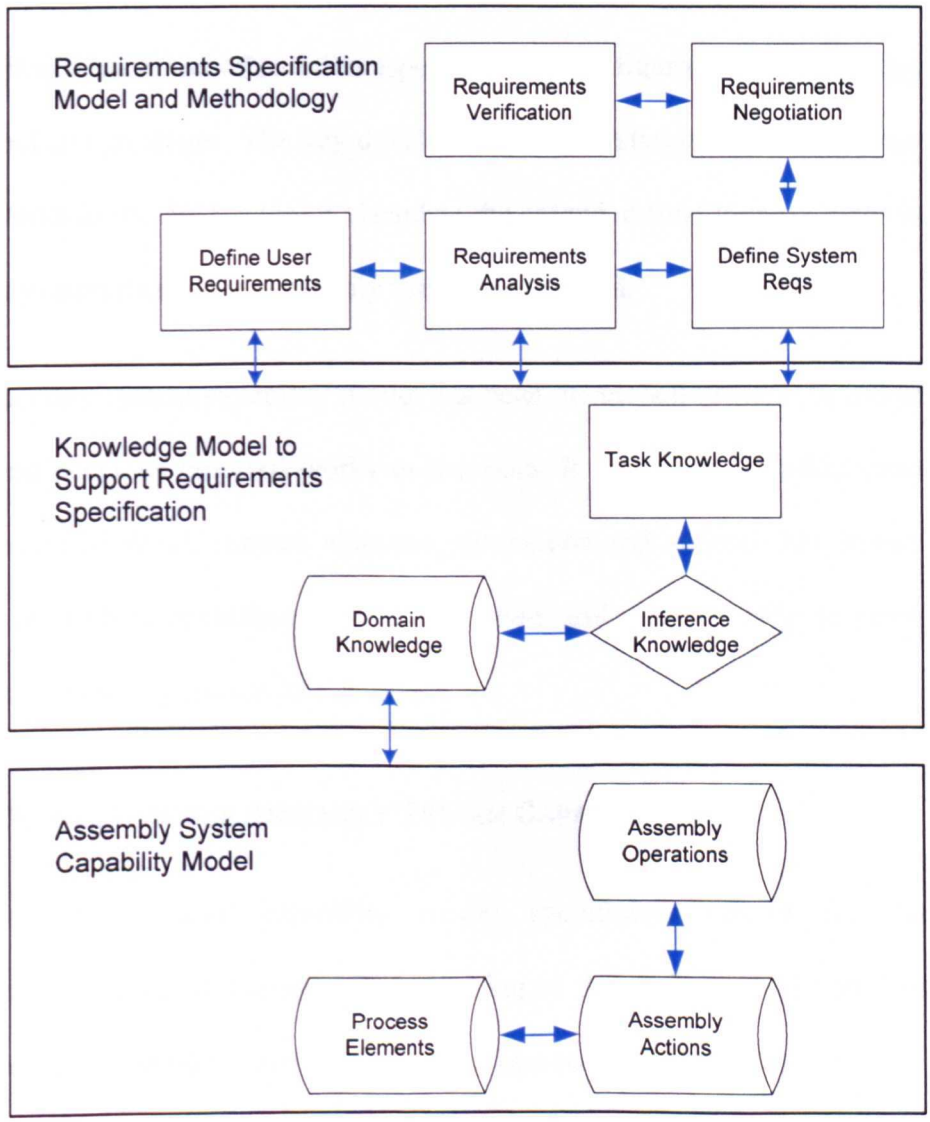


FIGURE 6-1: CONTEXT OF ASSEMBLY SYSTEM CAPABILITY MODEL

Whilst the requirements specification model and methodology concentrates on the overall process of defining the Reconfigurable Assembly System requirements, the

knowledge model is employed to support the requirements specification model and methodology. The role of the assembly system capability is to support the knowledge model by supplying assembly capability data to the domain knowledge schema (see Figure 6-1). Assembly operations, actions and process elements are supplied, which provide the building blocks for assembly task specification.

The assembly capabilities that combine to form assembly processes are essential to embedding system reconfiguration aspects into Reconfigurable Assembly Systems at the initial design stages. The key development that distinguishes this approach from other works in the field is that it considers the extended functionality inherent within assembly operations when assembly tasks are specified.

The assembly system capability model has been developed using both industrial use cases and previous reported works in the field. It describes assembly processes at three levels of detail; process elements, operations and actions. The impact of the model on system reconfiguration is discussed and a case study is presented to demonstrate the application of the model.

6.2 Overview of the Assembly System Capability Model

The assembly system capability model specifically covers the functional specification of the assembly system looking at the definition of part links, their corresponding assembly operations and the options that these will open for system reconfiguration. It aims to bridge the gap between current requirements and future requirements. In most cases current requirements can be specified to a reasonable degree of accuracy, as there is a ready product to provide assembly requirements for.

However, future requirements are much more difficult to define especially when the system user has little or no knowledge as to which products will be assembled in the future. Additional functionality needs to be built into a Reconfigurable Assembly System to address this imbalance and to maintain reconfigurability. The assembly system capability model supplies a channel through which this can be done under these conditions.

The capabilities of an assembly system are described in terms of an operations capability space, which includes the three aspects of an assembly system. This is illustrated in Figure 6-2.

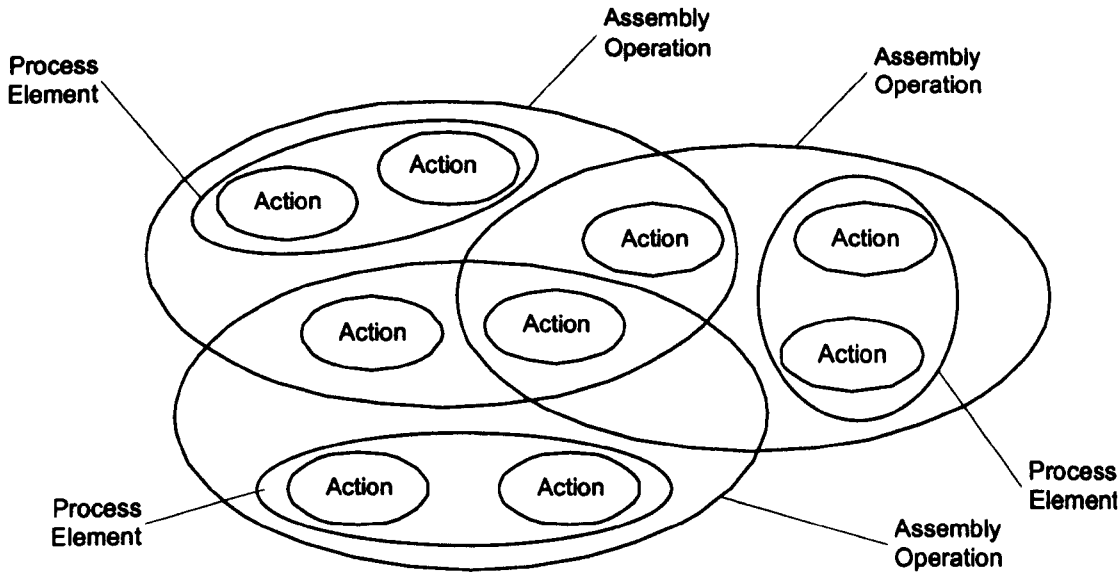


FIGURE 6-2: THE OPERATION CAPABILITY SPACE

All assembly operations are described as combinations of actions, each with their own specific parameters. Hence, Assembly Operations (O) are composed of Assembly Actions (A) such that:

$$O_i = \{A_j : A_j \in O_i, j=1....n\} \quad (1)$$

where O_i represents assembly operation i; and A_j represents assembly action j.

This relationship is defined by the characteristics of the individual processes and specifically their ability to perform the relevant operation.

Each operation is formally described with a name, motion set, work envelope, orientation and actions. These are key parameters that define specific methods for application. Each Action contains the parameters motion set and work envelope, which define the function performed by the assembly action. In mathematical terms an action belongs to an operation when:

$$A_k = \begin{cases} 1 & \text{if } A_k \in O_i \\ 0 & \text{if } A_k \notin O_i \end{cases} \quad k = 1.....n \quad (2)$$

The corresponding values for A_k are mapped through a table. The diagrammatic representation shown in Figure 6-2 is a simplified way of viewing the system capability space. It illustrates a 2-dimensional representation whereas in reality the space is multi-dimensional and can only be truly represented in matrix form.

The distribution of actions among different operations demonstrates the common and unique capabilities between different operations. This is formally represented by

process elements (E), which are unique clusters of actions that can be mapped to physical assembly equipment.

These are formally defined as

1. $E_i \cup E_j \equiv 0, \forall E_i, E_j \varepsilon \Re$

2. $A_k \in E_i \Leftrightarrow A_k \notin E_j, \varepsilon \Re$ (3)

where $\Re$ is the overall capability space and A_k is an assembly action. Process Elements do not overlap, that is they do not share actions. Hence, an action can belong to only one process element while the process element and its constituent actions can be part of a number of different operations. The introduction of process elements facilitates a more detailed understanding of the capabilities encapsulated in different operations and allows decision makers to outline the extra built-in capabilities in the equipment that can be utilised in future reconfigurations. For example, when considered independently, the "move" and "grasp" actions only provide movement and grasping capabilities; however, the combination of "move" and "grasp" actions provides a process element that can perform a pick and place function. Moreover each process element represents a different level of reconfiguration as highlighted in 6.6.

Describing the capabilities of a workstation using process elements helps indicate the extra effort required in terms of capability variation for system reconfiguration. An example of capability variation modelling using process elements for workstation reconfigurability is shown in Figure 6-3.

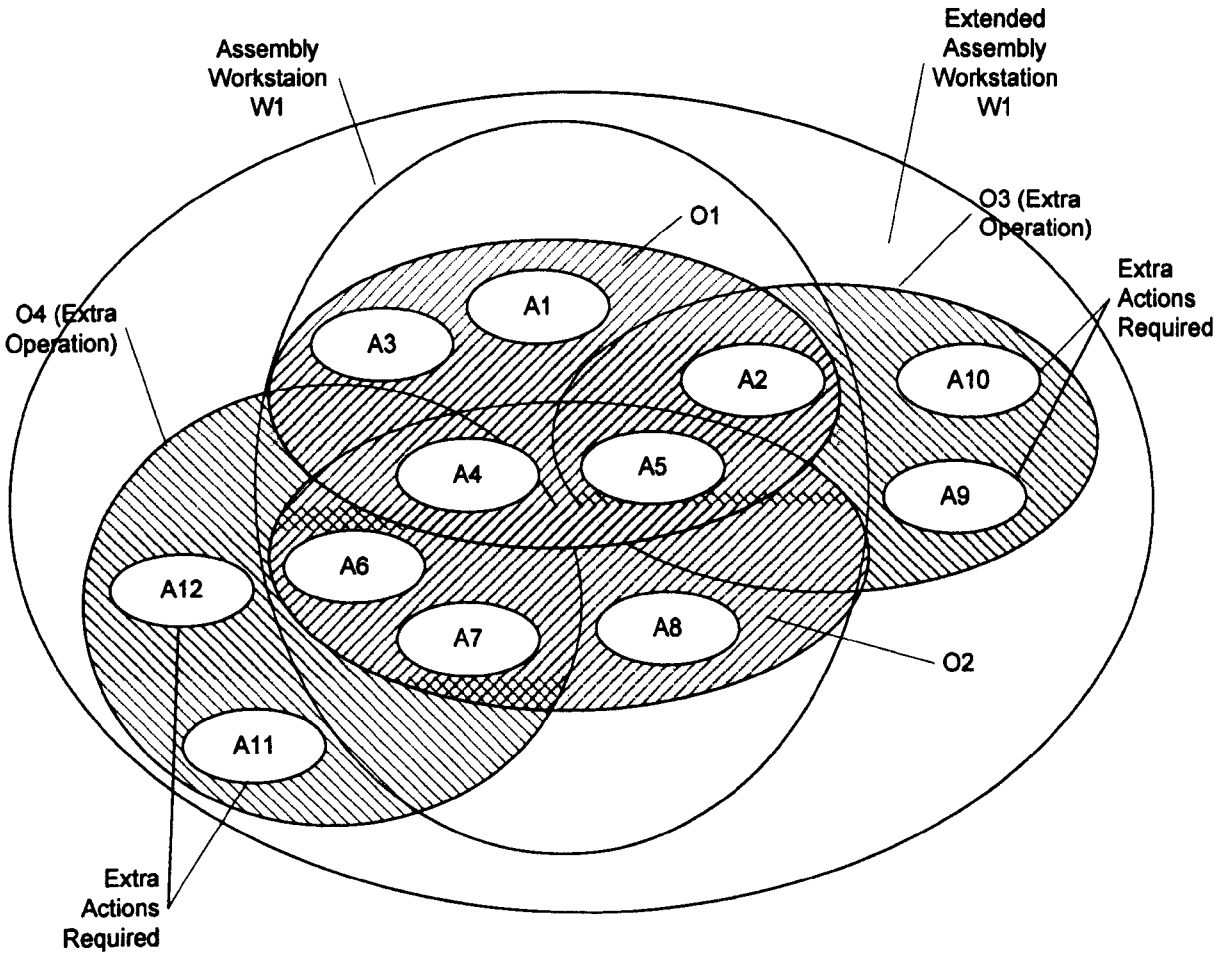


FIGURE 6-3: VARIATION OF CAPABILITIES FOR ASSEMBLY WORKSTATION RECONFIGURATION

The initial requirements are represented by workstation W1, which includes two operations O1, O2 and actions A1-A8, grouped into 3 E's. In terms of built-in reconfigurability the system capability description shows that there are 2 closely related operations O3, O4 which partially include actions already performed by the workstation. These represent the minimum reconfigurability potential of the workstation W1 for future expansion by including actions A9, A10 for O3 and A11, A12 for O4.

The impact of this on system reconfiguration is presented in 6.6. The various aspects of the model and their impact on assembly system reconfiguration are described in the remainder of this chapter.

6.3 CLASSIFICATION OF ASSEMBLY OPERATIONS

The assembly of a product is decomposed onto a set of tasks, where each task contains a set of operations. There are five groups of operations as per the classification in Figure 6-4.

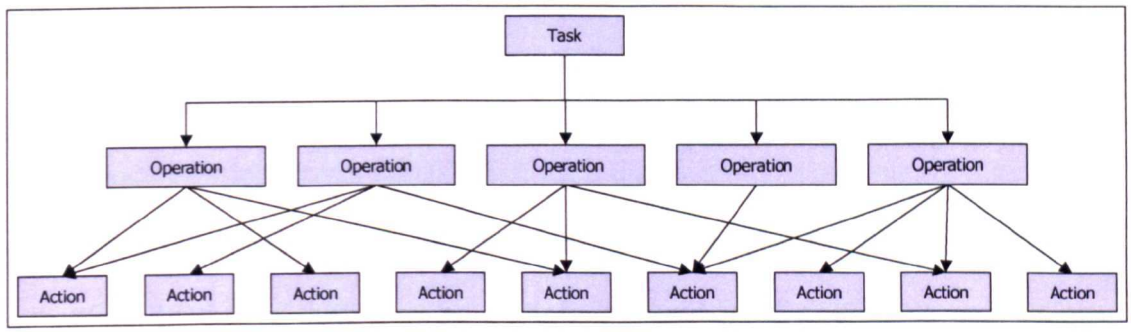


FIGURE 6-4: OPERATIONS IN AN ASSEMBLY TASK

There are five groups of operations as follows:

- **Feeding** encompasses all the actions that involve transporting the parts to the point of assembly, e.g., bowl feeding.

- **Joining** includes all the actions that result in two or more parts being joined together either permanently or non-permanently, e.g., snap fitting.

- **Testing** describes the actions that perform either a functional or non-functional test on the assembly, e.g., leak testing.

- **Material transfer** describes how parts are transferred between assembly workstations and/or assembly lines, e.g., by in line conveyor.

- **Special operations** are secondary operations that need to be performed for the primary joining to be completed. This would typically include tasks such as curing in electronic circuit board assembly.

Joining operations perform the key value adding function in an assembly task. The other four groups serve the joining operation and are referred to as ancillary operations. Only joining operations are considered in the operations capability model to keep the study concise. The joining operation is chosen to satisfy an assembly task and relevant ancillary operations are selected to complement that joining operation.

Each joining operation has its own set of actions that describe the properties of the operation. In theory there can be an infinite number of operations and actions. However, in this study the number has been limited to the following:

- **Adhesive bonding** involves two or more parts being glued together using some type of adhesive. The type of adhesive used depends on the nature of the materials being assembled and whether the bond is required to be permanent or semi-permanent.

- **Snap fit** occurs when two parts are assembled together semi-permanently. Parts slipping inside each other through slots and grooves secure the joint. Pressing one part in to release the other can disassemble the product.

- The **press in** operation is slightly different to the snap fit in that pressed in parts are held together due to the elastic properties of the two parts. For example a spring might be pressed into a gap between two parts.

- **Insert** operations involve the insertion of one part into a hole in the other. This is the simplest type of operation to perform.

- **Screw** operations use screws to hold one or more parts together. Different types of screws are used depending on the nature of the parts and the required type of joint. The type of screw is normally specified in the product design.

- The **rivet** operations consist of a permanent bond where a rivet is inserted into a hole between two or more parts and is then crimped at the ends so that it holds the parts together. The rivet must be destroyed to disassemble the parts.

- **Weld/Solder** operations use a substrate to join the two parts together. The substrate is dispensed to the joining point and melted. The two parts are then joined together while the substrate is in a molten state. A firm permanent bond is made when the substrate cools down and solidifies. The exception to this rule is laser welding, which does not use a substrate, but instead melts a small area on the parts and uses this to form the permanent bond. Welding and soldering are considered together due to their similar nature. Generally soldering only occurs in electronics assembly where assembled parts are held in place on a printed circuit board through soldered joints.

These operations are essentially sets of actions that have been put together to perform a specific function.

6.4 DESCRIPTION OF ASSEMBLY ACTIONS

Assembly actions represent the properties of assembly operations. Each action represents a single movement of a physical assembly performing entity and is defined by a set of attributes. Each attribute has some parameters that distinguish that particular instance of the action from others.

For requirements specification purposes, the attributes and parameters are stated generically so that they can later be used as a basis to perform a search for assembly equipment. The following sample actions are considered:

- The **support** action describes the payload that can be supported by an assembly system base. This base may cover both manual and automated workstations but the payload gives an indication of the weight of equipment and parts that can be supported by the base.

- The **dispense** action covers operations where a material has to be dispensed onto the assembly in a liquid state. This could be for lubrication purposes (e.g., greasing) or for a joining operation such as adhesive bonding or weld/solder operations. In each case the amount, density, nature of the liquid and size of the deposit are described.

- The **move** action covers any movement that is performed for the joining operation to take place. This can be either a point-to-point movement or

movement along a pre-defined path. In either case the speed, acceleration, accuracy and repeatability are described.

- The **index** action applies mainly to rotary indexing table (carousel) based assembly. This involves description of the angle through which the table should move, the payload that can be supported and the number of workstations that can be accommodated.

- The **apply heat** action is mainly relevant to weld/solder operations. The attributes here are profile of heat application and method of application.

- The **apply force** action describes the force that needs to applied to force fit a part into another. The magnitude and direction of the force is described. In this case the force being applied is different to that needed to merely move an object from one place to another.

- The **grasp/release** action forms the basis for insert, press in and snap fit operations where the part to be joined to the existing assembly is picked up from a predefined space and is then placed onto the product. This action takes into account the size and profile of the part and the forces that need to be applied to perform the operation.

- The **apply current** action is specific to laser welding operations where an electrical current is applied to generate a laser beam. The profile of the current has to be controlled to generate the correct precision for the joint.

Specific instances of the actions are instantiated by the assembly operations that they serve. The assembly actions need to be presented to the requirements specification model in a way that they can provide some functionality to accommodate system reconfiguration. This is the role of process elements.

6.5 DESCRIPTION OF PROCESS ELEMENTS

Assembly actions are grouped into process elements to provide the correct level of abstraction for system reconfiguration to take place. Each process element contains a set of actions that are unique to that process element. Since assembly modules are mapped directly to process elements, a change in process element will map to a change in assembly module and hence a system reconfiguration.

Defining operations in this sense is particularly useful when requirements are defined for new Reconfigurable Assembly Systems. The number of process elements an operation includes indicates the scope for reconfigurability that operation brings to the assembly system. The following algorithm has been developed for grouping the assembly actions into process elements.

$$\sum_{i=1}^{n} O_i \in E_i$$

$$\sum_{k=1}^{n} A_k \in O_i \Rightarrow A_i \in E_i$$

$$\sum_{k=1,j=1}^{n} A_{kj} \in O_k \cap O_j \Rightarrow A_{kj} \in E_{kj}$$

$$\sum_{k=1,j=1,l=1}^{n} A_{kjl} \in O_k \cap O_j \cap O_l \Rightarrow A_{kjl} \in E_{kjl}$$

The steps involved are as follows:

Step 1: For all O_i, create resource Element E_i

Step 2: Find all A_k: $A_k \in O_i$, put in E_i

Step 3: Find all A_k: $A_k \in O_i \cap O_j$, put in E_{ij}. If E_{ij} already exists. If E_{ij} does not exist then create E_{ij}.

Step 4: Find all A_k: $A_k \in O_i \cap O_j \cap O_k$, put in E_{ijk}. If E_{ijk} does not exist, create E_{ijk}.

Step n: Continue until all actions and operations have been defined within process elements

Using this algorithm, options for reconfigurability for the operation $O_j = E_j$, E_{jl}, E_{jk}, E_{jkl} are O_l and O_k as process elements are shared between these assembly operations. The addition of process element E_l will expand the system capability from O_j to $O_j \cap O_l$. Moreover, the level of reconfigurability inherent to an operation is evaluated by assessing the number of process elements within that operation.

If the goal is to maximise reconfigurability then the option that generates the highest number of alternative operations is required. This is also the operation that includes the most number of process elements.

$$\text{Max } R = \text{Max } (E_{ix} \in O_i \cup O_x) \qquad (5)$$

Where E_{ix} is the number of process elements that belong to the chosen operation O_i and the Alternative Operation O_x

Assembly systems can be developed that consider future requirements before the system is designed making the system adaptable to future change at little cost with the aid of the assembly system capability model. Updates to the assembly system capability model are easy to implement as developments in assembly operation technology can be integrated into the model by adding new operations and actions.

The result of processing the clustering algorithm is the assignment of process elements to three distinct types of system reconfiguration. This is based on the notion that Reconfigurable Assembly Systems are constructed from layers and that assembly modules belong to one of the three layers. When a system reconfiguration takes place, one of these layers is reconfigured as one or more modules are exchanged or modified.

6.6 LEVELS OF RECONFIGURATION

The levels of system reconfiguration correspond to the three types of equipment change that can take place on a Reconfigurable Assembly System. System, process and product level system reconfiguration is undertaken according to changes in process elements.

6.6.1 Configuration at System Level

System level reconfiguration revolves around changes made to the base layer of the Reconfigurable Assembly System. This is the platform upon which the assembly system is mounted. Manual cells, rotary indexing tables (carousels) and pallet based conveyor systems are examples of Reconfigurable Assembly System base layers.

The two action types involved here are the index and support actions. Instances of these actions in general form their own individual process elements and serve many operations. This can be related to Figure 6-5 where the support and index actions form a process element that is used for gluing, insertion and snap fit operations.

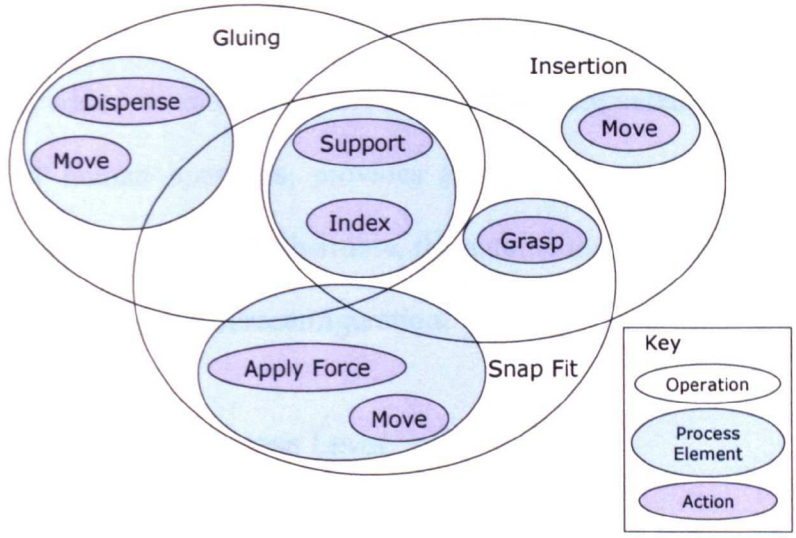


FIGURE 6-5: ACTIONS IN SYSTEM LEVEL RECONFIGURATION

System level reconfiguration is desirable if one or more of the following conditions are true:

1. The number of assembly workstations is likely to change as a result of a change in product design.

2. One or more products with very different parts are likely to be assembled on the same assembly line.

3. There are significant fluctuations in output volumes.

This type of system reconfiguration has the greatest limitations as it is time consuming and costly to change for example from a carousel system to a pallet based

conveyor system. If system level reconfiguration occurs, equipment on other layers can still be reused although this may be time consuming and costly.

Another application of system level reconfiguration is the reconfiguration of an existing base by adding or removing individual workstations. This may occur when a product is redesigned so that some parts are removed. Interchanging manual stations, which can be added or modified more easily than automated stations due to the capabilities of human operators, provides the greatest practical opportunity for system level reconfiguration. Furthermore, the automation of manual workstations is an application of system level reconfiguration.

6.6.2 Configuration at Process Level

Process level reconfiguration deals with changes in the operations performed within a Reconfigurable Assembly System. In essence this is based around the assembly movements that are performed on the assembly line together with ancillary actions that support the specific assembly operations.

Process elements at this level cover the move actions together with some actions specific to some operations such as dispense and grasp as shown in Figure 6-5.

Process level reconfiguration demonstrates the ability of Reconfigurable Assembly Systems to be reconfigured to perform different operations by adopting additional process elements. For example, as long as the global parameters of the move actions remain consistent, operations can be reconfigured to perform insertion, snap fit and force fit operations.

Moreover process level reconfiguration takes place when assembly processes change without significant changes in production volumes or the number of assembly workstations: the operations being performed at the respective assembly workstations change.

A practical application of this is to change the manipulator of an already selected robot configuration to perform a different operation. For example a SCARA robot can perform both insertion and screwing operations, depending on the manipulator that is mounted on it. The reconfiguration is needed if for example a product design is modified to comply with DFA techniques whereby snap fit joints replace screw joints.

6.6.3 Configuration at Product Level

Product level reconfiguration refers to the ability of the Reconfigurable Assembly System to adapt and assemble several products that have similar parts. In this case assembly actions and operations remain the same. However the parameters that determine the product specific aspects of the assembly are adjusted for specific application to the product.

There are two scenarios that use product level reconfiguration:

1. Minor design changes are made to product parts without causing a change in the actual assembly operation employed.

2. Similar products with similar parts are assembled on the same line within a similar cycle time.

In practice product level reconfiguration means adjusting assembly fixtures and machines to assemble slightly different parts.

6.7 Specification of Requirements for Assembly System Reconfiguration

Assembly system reconfiguration takes place due to changes in the assembly requirements for a product either through the introduction of new products to be assembled on the Reconfigurable Assembly System or design modifications to existing parts. It is important to understand the levels of reconfiguration so that reconfigurability can be built into an assembly system when it is designed.

Requirements for system reconfiguration are established based on the levels of reconfiguration and the occurrence of each type of reconfiguration. These are extracted from the system user as part of user requirements specification so that reconfigurability forms part of the design requirements for the system. These are the requirements for future modification as briefly outlined in section 4.2.

This part of the requirements model is constructed as a set of questions about the expected use of the system:

1. Will the assembly system assemble only one product throughout its life?
2. Are the products likely to change during the system lifespan?
3. Are the number of assembly stations likely to change as a result of the product change?

4. Will products with similar part sizes be assembled on the same assembly line?

5. Are assembly operations likely to change?

6. Are there significant ramp-up effects to take into account?

7. Will there be significant fluctuations in production volumes?

Answers to these seven questions are needed to determine the type of reconfiguration that is needed. Table 6-1 summarises the characteristics of the different levels of reconfiguration.

Level of Reconfiguration	Scenarios
System	Many different products with different operations and fluctuating volumes
Process	Different assembly operations but performed in similar volumes
Product	Similar products with same assembly processes in similar volumes.
None (fixed automation)	No changes

TABLE 6-1: SUMMARY OF RECONFIGURATION LEVEL CHARACTERISTICS

The characteristics of reconfiguration levels must be captured to determine which level of reconfigurability to build into a Reconfigurable Assembly System. Answers to the questions above are compared to the level of reconfiguration characteristics and requirements for the level of reconfigurability to build into a Reconfigurable Assembly System are established. This is implemented through the requirements specification knowledge model presented in section 5.

6.8 Assembly System Capability Model – Application Case

Implementation of the assembly system capability model is demonstrated using an industrial example. The example is based on the assembly process capabilities derived for a drug delivery system assembly for a multinational manufacturer; the system design and development was conducted by an SME[‡].

Assembly of the drug delivery device involved the snap fit and insertion of four separate parts, of which one contained a subassembly. A breakdown of assembly operations for each part is presented in Table 6-2 under the "parts", "operations" and "original actions" columns. The device was made of plastic parts and delivered dry powder medication to the patient through the oesophagus.

Part	Operations	Original Actions	New Actions	Change Needed
Base Subassembly	Insertion	Move, Grasp	Move, Grasp	Variation in Weight of Part
Body Half Top	Snap Fit	Apply Force, Move, Grasp	Apply Force, Move, Grasp	Variation in Weight of Part
Mouthpiece	Snap Fit	Apply Force, Move, Grasp	Apply Force, Move, Grasp	New Part
Outer Case	Snap Fit	Apply Force, Move, Grasp	Apply Force, Move, Grasp	Variation in Weight of Part
Digital Counter	Insertion	Move, Grasp	Move, Grasp	New Workstation

Table 6-2: Breakdown of Assembly Operations for Drug Delivery Device

[‡] Company details cannot be disclosed to preserve commercial confidentiality

The assembly sequence for the device was as follows:

- The base subassembly is loaded and held into place.
- The body half top is snap fitted on top of the base subassembly.
- The mouthpiece is inserted onto the device from the side
- The outer case is snap fitted to protect the device.

One of the key requirements was to develop an assembly system that could assemble the product in a cycle time of less than 4 seconds. Another key requirement was that the assembly system had to be able to assemble two similar products for as it was envisaged that a modified version of the device would be assembled in the future. The modified device was going to be made of aluminium, which is a material that is liable to scratch easily and this had to be prevented to maintain a shiny surface finish. In addition to this a new mouthpiece design would be used, which would make the drug delivery more efficient. In addition to this a digital counter was going to be added to show the number of doses left in the device.

As a result of the extra requirements the systems integrator had to consider a variety of possible system modifications and their implications on the overall cost and system performance. Requirements for these modifications are listed in Table 6-3.

The implications of the part changes are described in terms of changes in operations and process elements. These affected the workstations where the changed parts were assembled. Furthermore an extra workstation was included to assemble the digital counter.

Complications caused by the modification of the device material were countered by the suggestion of having the parts supplied with a protective film covering the scratchable surfaces.

Part	Potential Modification
Base Subassembly	More durable case made of aluminium
Body Half Top	More durable case made of aluminium
Mouthpiece	Modified version with new dimensions for more effective drug delivery
Outer Case	More durable case made of aluminium
Digital Counter	New Part

TABLE 6-3: POSSIBLE MODIFICATIONS TO DRUG DELIVERY DEVICE DESIGN

The "new action" and "change needed" columns in Table 6-2 show the changes required to assemble the new modified device. Analysis of Table 6-2 shows that the original actions and new actions were similar. From this it was deduced that no Process Level Reconfiguration was required. However Product Level Reconfiguration was required as there were changes to the parameters of the assembly actions.

Furthermore System Level Reconfiguration was needed as the addition of the digital counter that an additional workstation was required to perform this function. Analysis of the operations showed that although Process Level Reconfiguration was not needed, it could still be easily accommodated as the insertion and snap fit operations contain common actions. The main difference is that the snap fit operation involves extra force to fix the part in place semi-permanently.

Operation	Actions	Process Elements
Adhesive Bonding	Dispense, Move, Grasp, Hold	D1, M1, G1
Snap Fit	Move, Hold, Apply Force	M1, F1
Screw	Move, Hold, Apply Torque	M1, T1
Rivet	Move, Hold, Apply Force	M1, F2
Press In	Move, Hold, Apply Force	M1, F2
Insert	Move, Hold, Grasp	M1, G1
Weld/Solder	Dispense, Move, Hold, Apply Heat/Current	D2, M1, A1

TABLE 6-4: SAMPLE CLASSIFICATION OF OPERATIONS

The classification of operations in Table 6-4 defines the options for Process Level Reconfiguration that can be adopted if extra capabilities are required in the future. The commonality of process element M1 indicates that these processes can be reconfigured into any of the processes in the table, by adding assembly modules that can perform the assembly capabilities highlighted by the additional process elements.

6.9 CHAPTER SUMMARY

A new Assembly System Capability Model has been created that captures the inherent properties of assembly processes so that they can be mapped to assembly modules. Assembly operations are decomposed into assembly actions, which are grouped into process elements using a clustering algorithm. This provides the correct level of abstraction for mapping assembly modules to assembly processes.

The key development here is the discovery of common properties of assembly operations indicating the ability of assembly processes to be reconfigured at different levels of modularity. As a result several levels of reconfiguration have been defined:

- Product level reconfiguration means only a modification to the process parameters is made;
- Process level reconfiguration means that the assembly workstation is reconfigured to perform a different assembly process;
- System level reconfiguration implies a fundamental shift in the assembly system configuration, for example the addition or removal of assembly workstations or the conversation of a manual workstation into an automatic one.

Application of the assembly system capability model has been demonstrated through the case study of a drug delivery device assembly.

7 SYSTEM IMPLEMENTATION AND METHODOLOGY VERIFICATION

7.1 INTRODUCTION

The research outcomes are illustrated through application to an industrial scenario. This scenario charts the requirements specification process for a Reconfigurable Assembly System for the assembly of a car glove box latch.

The scenario is described with reference to implementation in the requirements specification pilot environment that has been created as part of this research. The pilot environment is described with reference to the three levels of knowledge stated in section 5.

User requirements specification and system requirements specification are described for the Southco glove box latch assembly. User requirements for system reconfiguration are specified and the effect of these on the system requirements is analysed.

7.2 DESCRIPTION OF PILOT ENVIRONMENT

A pilot decision making environment has been developed to demonstrate requirements specification for Reconfigurable Assembly Systems (see Figure 7-1). The environment interfaces with the web through an apache web server, which is kept behind a firewall to maintain system security. Web pages are displayed to the user as HTML pages and the server exchanges messages and code with a JSP Servlet, which calls different program modules represented as Java Beans and Java

Objects. These provide the task knowledge functionality from the requirements specification model.

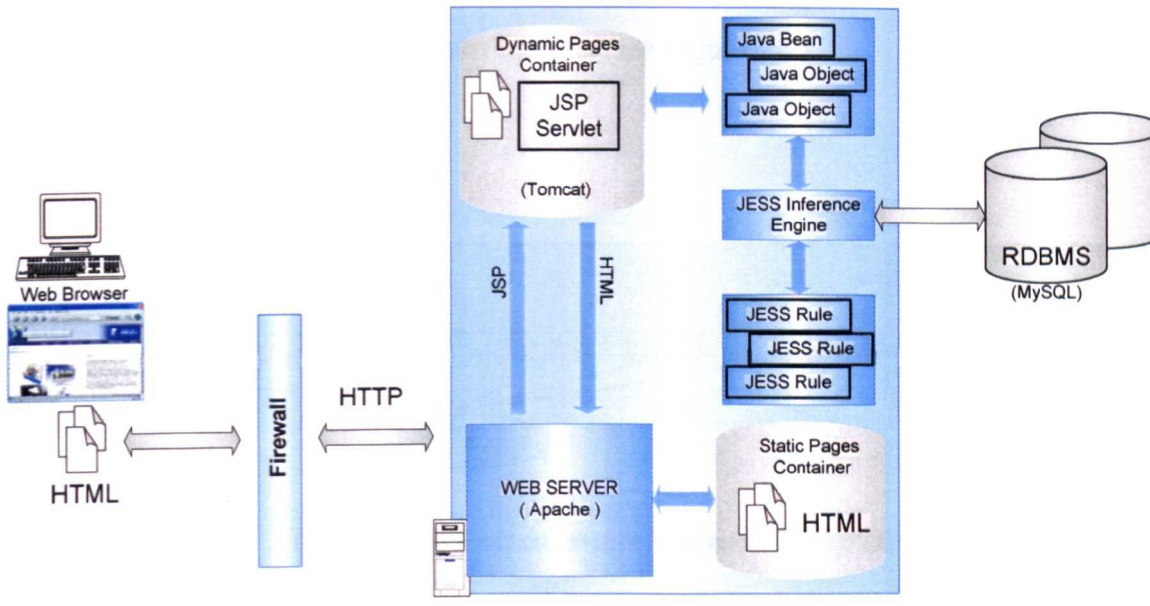


FIGURE 7-1: OVERVIEW OF PILOT ENVIRONMENT FOR REQUIREMENTS SPECIFICATION OF RECONFIGURABLE ASSEMBLY SYSTEMS

The JESS inference engine supports the Java Beans and Java Objects. This manages the execution of JESS rules and provides the connection between the domain knowledge, which is stored in a MySQL Relational Database, and the task knowledge.

The user interface for the environment is covered from two perspectives: the system user perspective, where user requirements are defined; and the systems integrator perspective, where user requirements are converted into system requirements. These two perspectives have been accommodated through two sub-environments.

7.2.1 User Requirements Specification Sub-Environment

The user requirements specification sub-environment includes the specification of business constraints, product data, data on the parts that make up the product and how the parts are related through part liaisons (see Figure 7-2).

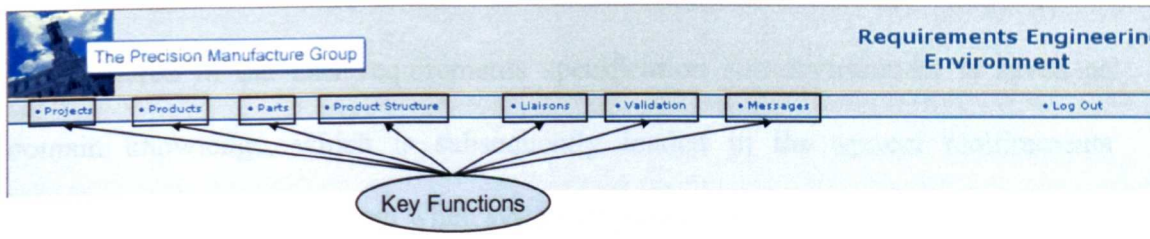


FIGURE 7-2: KEY FUNCTIONS OF USER REQUIREMENTS SPECIFICATION SUB-ENVIRONMENT

Clicking on each of the menu items displayed in Figure 7-2 results in the appearance of a data gathering form. These forms prompt the system user to enter data under the respective headings. A brief description of the contents under each heading is given below:

- Projects – project requirements are specified to populate the projects domain.
- Products – product characteristics are stated to populate the products domain.
- Parts – each part within the assembly described to populate the parts domain.
- Product Structure – parts that are described for the product are placed into a tree structure whereby subassemblies can be distinguished and subsequently treated as independent products assuming that each subassembly can be assembled by a subsystem.
- Liaisons – links between the parts are stated together with the link characteristics to populate the liaisons domain.

- Validation – clicking here results in the execution of requirements analysis on the system user side, whereby missing and ambiguous requirements are uncovered and the system user is prompted to supply these before the requirements can be submitted to a systems integrator.
- Messages – once the user requirements have been validated, they are sent in a message to systems integrators for specification of systems requirements.

Data entered in the user requirements specification sub-environment is saved as domain knowledge, which is subsequently loaded in the system requirements specification sub-environment when system requirements are specified.

7.2.2 System Requirements Specification Sub-Environment

The system requirements specification sub-environment guides the systems integrator through the system requirements specification process. Figure 7-3 illustrates the homepage including menu items for system requirements specification.

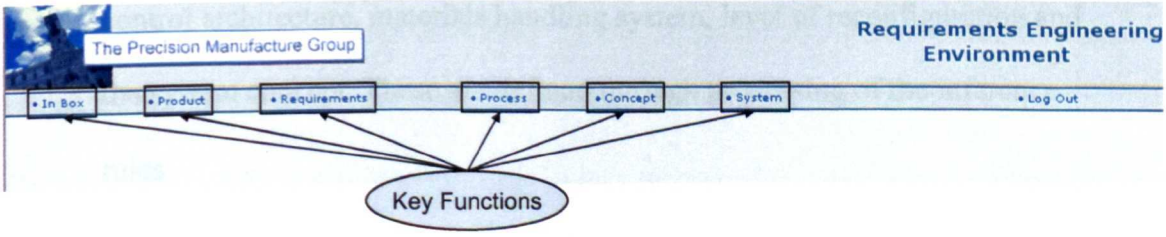


FIGURE 7-3: KEY FUNCTIONS OF SYSTEM REQUIREMENTS SPECIFICATION SUB-ENVIRONMENT

Each menu from the homepage is navigated and the relevant forms are filled in by the systems integrator to derive the system requirements specification. Each menu item is explained below:

- Inbox – projects that are sent to the systems integrator arrive here. The message from the system user and the user requirements domain knowledge is received by the systems integrator.
- Product – an overview of the product, with its parts and various diagrams are viewed by clicking here.
- Requirements – non-functional requirements, such as spatial constraints and services available for the assembly system are viewed by clicking here.
- Process – parts are removed from the product structure by the systems integrator to create a disassembly sequence, which is then reversed to create an assembly task sequence. Hence, assembly tasks are derived for the product assembly where each task includes assembly operations and part feeding requirements. Assembly operations and part feeding methods for each task are derived automatically using the inference rules.
- Concept – system concept specification includes the specification of the control architecture, materials handling system, level of reconfiguration and the system concept. These are defined through processing of the inference rules.
- System – business requirements for the project are verified where costs and timescales are considered for each assembly task and elements of the system concept. Checks are performed using inference rules to verify that the costs and timescales are within the budget and timescale requirements stated in the user requirements specification.

Whilst user requirements specification is mainly a data entry process, system requirements specification is concerned with processing the data stored in the domain

knowledge base as a result of user requirements specification. Further explanation of the pilot environment is not given here as the industrial case study provides a much more vivid insight into how the pilot environment is used.

7.3 VERIFICATION SCENARIO – SOUTHCO LTD. GLOVE BOX LATCH ASSEMBLY

The Southco glove box latch assembly consists of 5 parts that are assembled in sequence to form the glove box latch mechanism for a Ford Focus (B Car). The parts and assembly process are outlined in Figure 7-4.

The base (housing) is held in place and the pawl and torsion spring are snapped into a slot in the housing. The lockplate is then inserted into a groove on the housing from the side after which the compression spring is inserted into a slot on the top of the housing. The assembly is finished with the paddle being snapped into place on top of the housing, with the compression spring underneath the paddle.

User requirements for a Reconfigurable Assembly System to assemble this product have been entered into the pilot environment developed to demonstrate the research outcomes. This section describes the elicitation of user requirements and the derivation of system requirements from these user requirements. User requirements for a similar product to be assembled on the same assembly system are also parsed through the knowledge-based system to derive system requirements for reconfiguration. This is a glove box latch for the Land Rover (C Car), which is proposed for assembly on the same line.

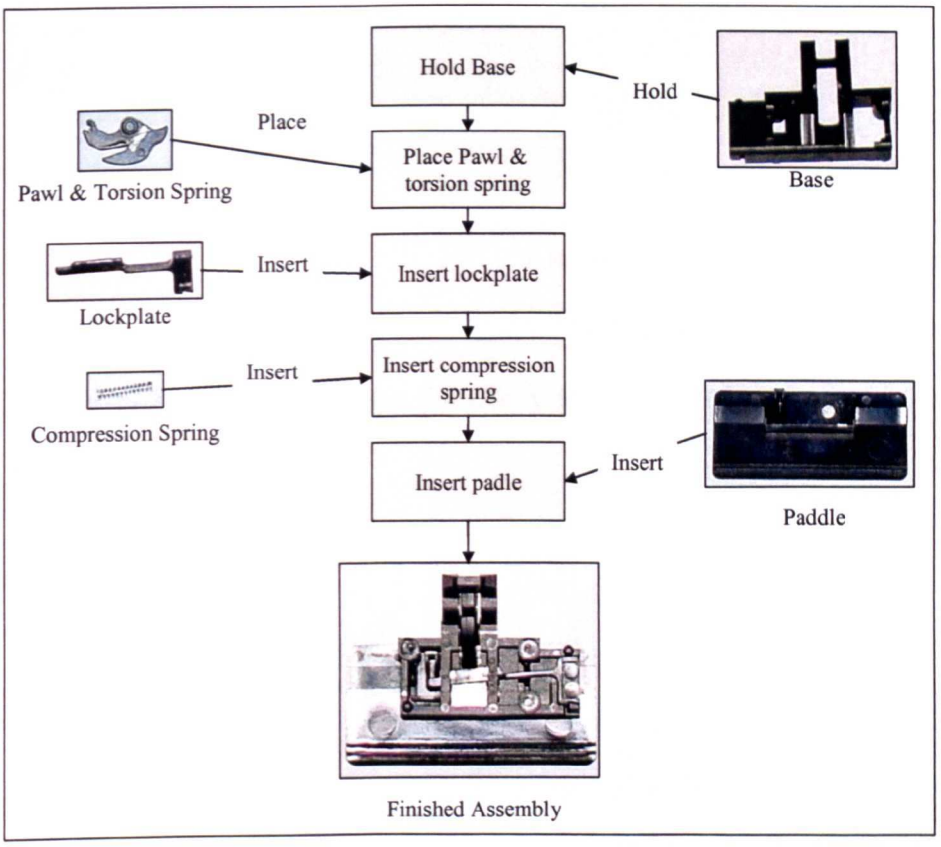


FIGURE 7-4: SOUTHCO GLOVE BOX LATCH ASSEMBLY

7.3.1 *User Requirements Specification for New Reconfigurable Assembly System*

User requirements specification (Figure 7-5) captures the user requirements for the project. Each part of the process is listed along the top of the requirements engineering environment. The user begins by clicking on the "projects" menu and navigates through each of the subsequent menus until "validation" of the data is performed and the data entered in the respective forms is then sent to the system requirements specification sub-environment.

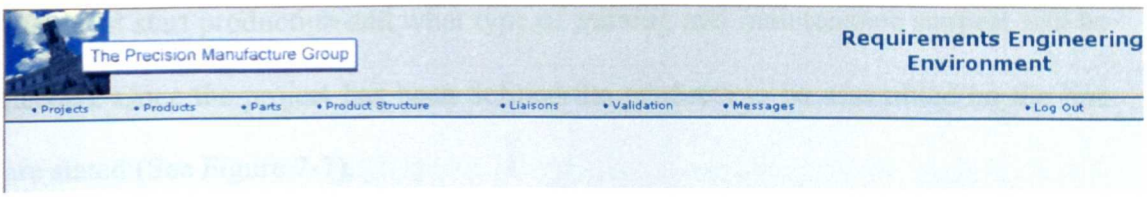


FIGURE 7-5: USER REQUIREMENTS SPECIFICATION HOMEPAGE

The "Projects" function aims to capture the business requirements for the project. The business requirements for the Glove Box Latch (SCO2) project can be seen in Figure 7-6.

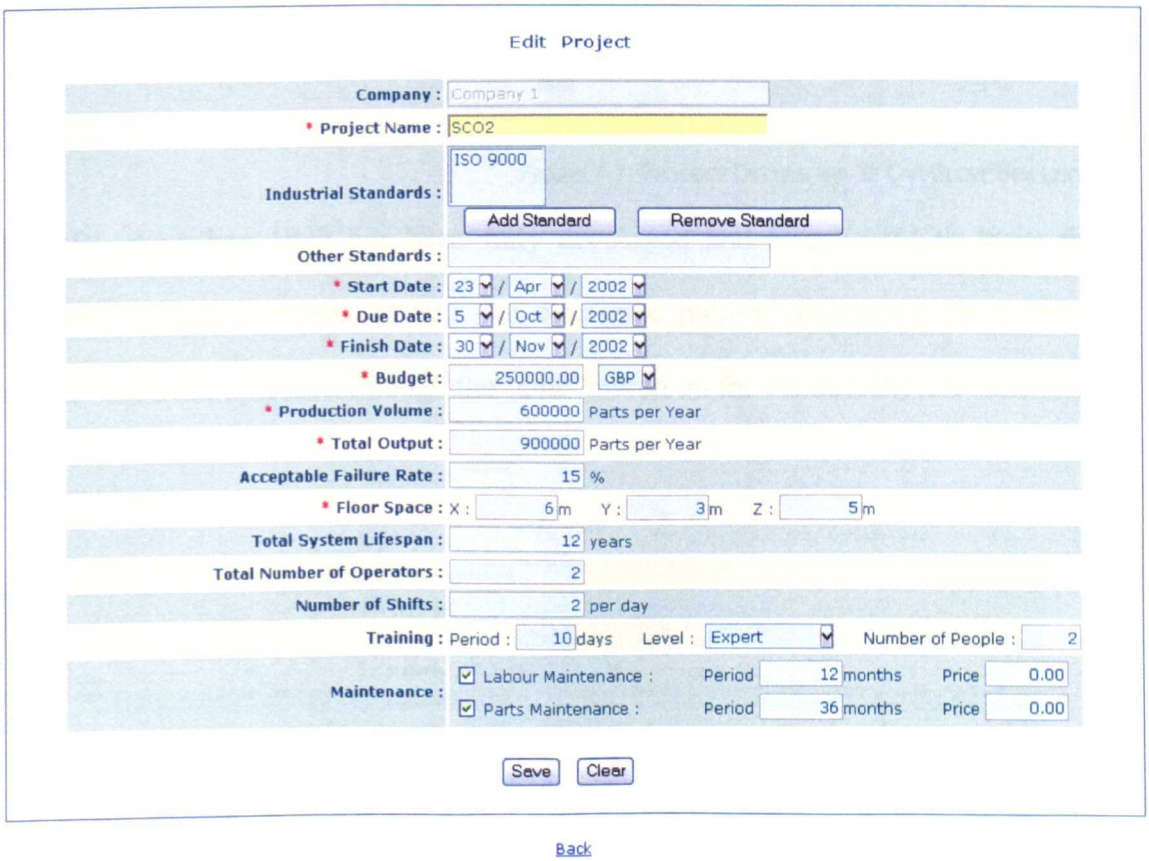


FIGURE 7-6: BUSINESS REQUIREMENTS FOR SCO2 PROJECT.

These are the general guidelines that must be followed for the acceptance of the tender. The aim here is to clarify how much money is available for the project, the operator support is available and the overall constraints such as when the assembly

line must start production and what type of training and maintenance support will be needed. Once the project has been defined the products to be assembled on the line are stated (See Figure 7-7).

FIGURE 7-7: PRODUCT DETAILS FOR 'B' CAR GLOVE BOX LATCH

The 'B' glove box latch has been fully developed and is expected to leave the assembly system in boxes of 500. An electronic picture illustrating the finished product has been uploaded through the "add" button on the Product Picture function.

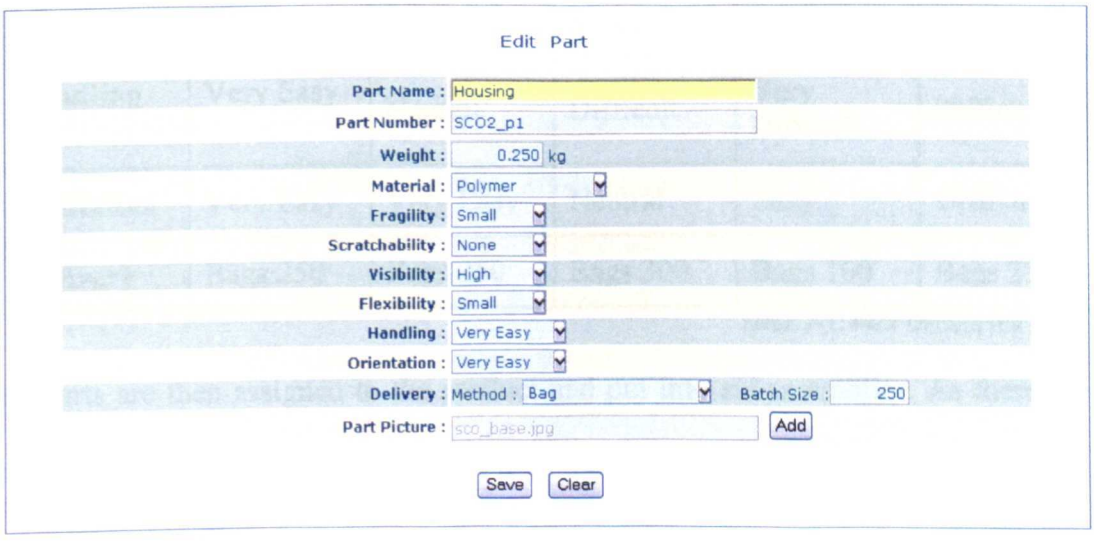


FIGURE 7-8: HOUSING PART FOR SCO2

The next step is to describe the parts that are to be assembled (see Figure 7-8). The part named "Housing" is described, which weighs 250gms and is made of a polymer plastic. The DFA characteristics for the part are described and the method of part supply is stated, i.e., that housings will be supplied in bags of 250. A picture of the part has also been uploaded. Similarly, all the parts have been described in this way. A summary of the parts data is presented in Table 7-1.

Part Name	**Housing**	**Lockplate**	**Pawl**	**Spring**	**Paddle**
Part Number	SCO2_p1	SCO2_p2	SCO2_p3	SCO2_p4	SCO2_p5
Weight (g)	250	150	100	5	200
Material	Polymer	Polymer	Polymer	Non-ferrous metal	Polymer
Fragility	Small	Medium	Medium	High	Medium
Scratchability	None	None	Small	None	Small
Visibility	High	Very High	High	High	Medium
Flexibility	Small	Medium	Small	Very High	Small
Handling	Very Easy	Neutral	Difficult	Very Difficult	Difficult
Orientation	Very Easy	Very Easy	Neutral	Easy	Difficult
Delivery	Bags 250	Bags 250	Bags 300	Bags 100	Bags 250

TABLE 7-1: PART DETAILS FOR SCO2

The parts are then assigned to the product and put into subassemblies. As there are no subassemblies involved here only the parts are assigned and a summary page is created that describes the whole product together with its parts.

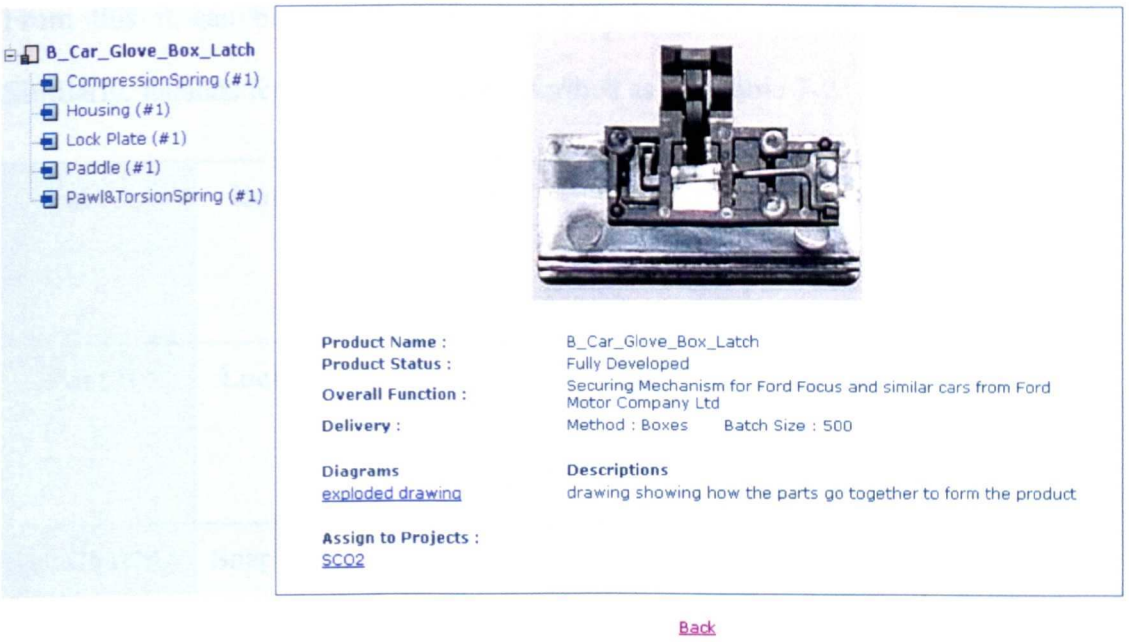


FIGURE 7-9: PRODUCT STRUCTURE FOR SCO2 B CAR GLOVE BOX LATCH

The part tree is shown on the left hand side, whilst the summary details for the product are displayed on the right. Clicking on the hyperlinks for "Projects" and "Diagrams" allows the user to have access to the data. Thereafter the part liaisons are formalised. The result is shown in Figure 7-10.

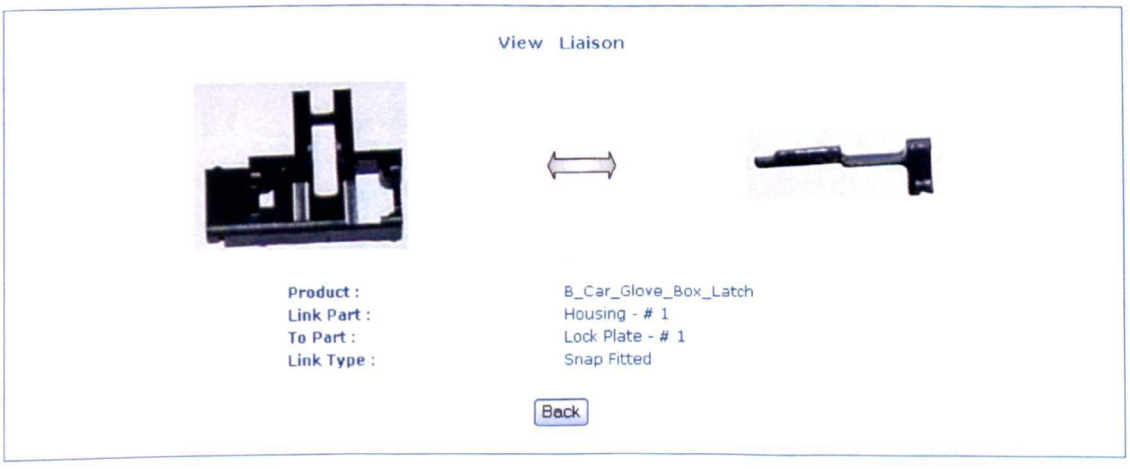


FIGURE 7-10: PART LIAISON INFORMATION FOR HOUSING AND LOCKPLATE RELATIONSHIP

From this it can be seen that the lockplate is "Snap Fitted" into the housing. Similarly, liaisons for all the parts are described as per Table 7-2.

Part A	Housing	Housing	Housing	Housing	Compression Spring
Part B	Lockplate	Pawl	Compression Spring	Paddle	Paddle
Relationship	Snap Fitted	Snap Fitted	Inserted	Snap Fitted	Inserted

TABLE 7-2: PART LIAISON CHARACTERISTICS FOR SCO2 B CAR GLOVE BOX LATCH

Once the part liaisons have been defined the data for the project is verified automatically by pressing the product verification link on the main menu. The data can then be sent for system requirements specification together with the information shown in Figure 7-11. The user requirements are sent to system integrator(s) with a short message.

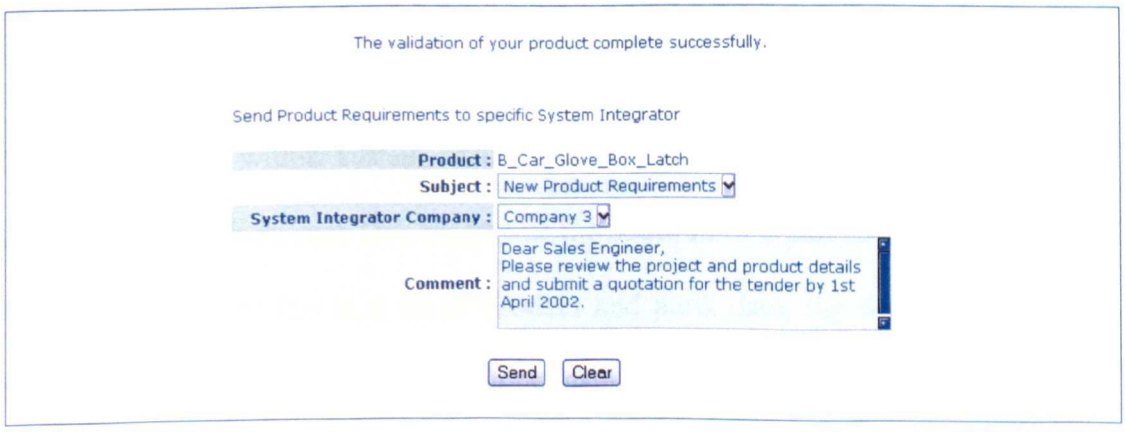


FIGURE 7-11: VERIFICATION SCREEN FOR SCO2 B CAR GLOVE BOX LATCH

This completes the user requirements specification and the data is sent for specification of system requirements.

7.3.2 System Requirements Specification for New Reconfigurable Assembly System

Whilst the user requirements specification deals with the product model, the system requirements specification concentrates on the process model for assembling the product. This is reflected by the menu structure of the System Requirements Engineering Environment (see Figure 7-12).

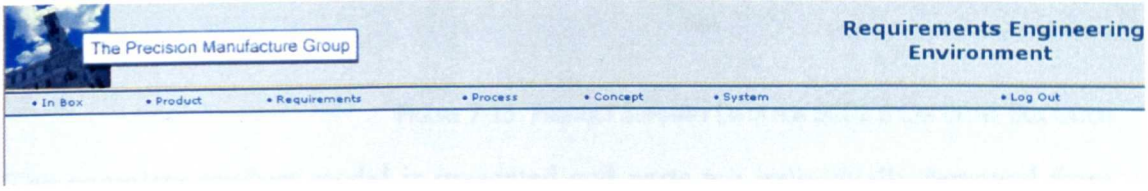


FIGURE 7-12: HOMEPAGE FOR SYSTEM REQUIREMENTS SPECIFICATION

The Inbox link contains messages passed on by the user requirements specification. Product information is accessed by clicking on the "Product" menu (see Figure 7-13). This is a page of static data about the product. Similarly pages about each individual part can be seen by clicking on the parts on the product tree on the left hand side of the page.

The core activity within system requirements specification is the specification of assembly tasks, where each task represents the addition of a part to the assembly. As the starting point of this is a set of product and parts data, the task sequence and structure needs to be derived. This is done manually through analysis of disassembly sequences.

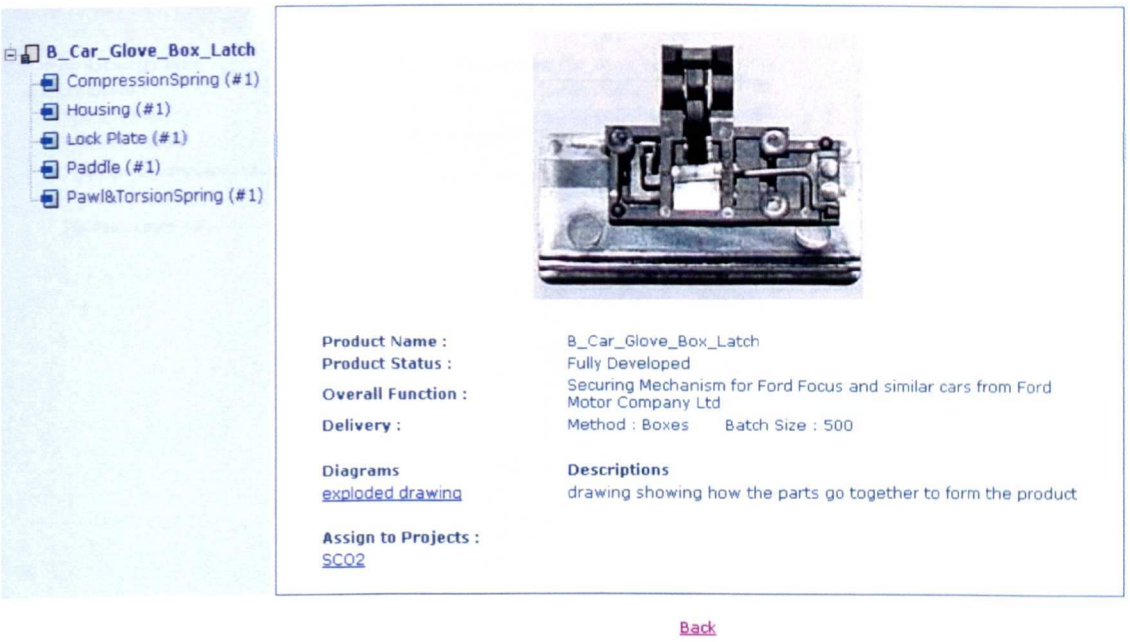


FIGURE 7-13: PRODUCT SUMMARY DATA FOR SCO2 B CAR GLOVE BOX LATCH

The complete product model is presented and parts are individually removed from the product to form separate states. Each part removal is a disassembly step, which then equates to an assembly task when reversing the process to represent the product being assembled. The steps are illustrated in Figure 7-14.

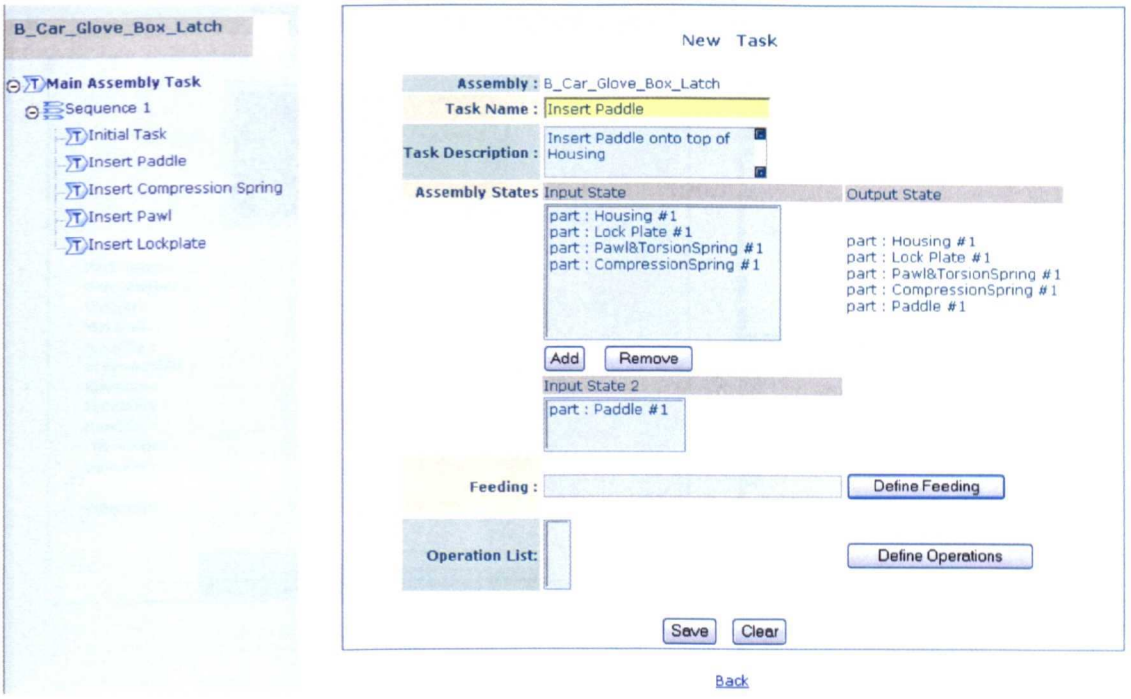


FIGURE 7-14: NEW TASK SPECIFICATION FOR SCO2 B CAR GLOVE BOX LATCH

The task description illustrated in Figure 7-14 shows the addition of the Paddle to the finished product. The task is given a name and a description. The two input states describe the two entities that are going to be put together, i.e., the nearly finished product and the paddle.

As each task represents the addition of a part to the assembly, this includes the assembly operations that are required to add this part together with the part feeding method. The recommended feeding method for each part is stated based on inference rules in the knowledge model. This is illustrated in Figure 7-15.

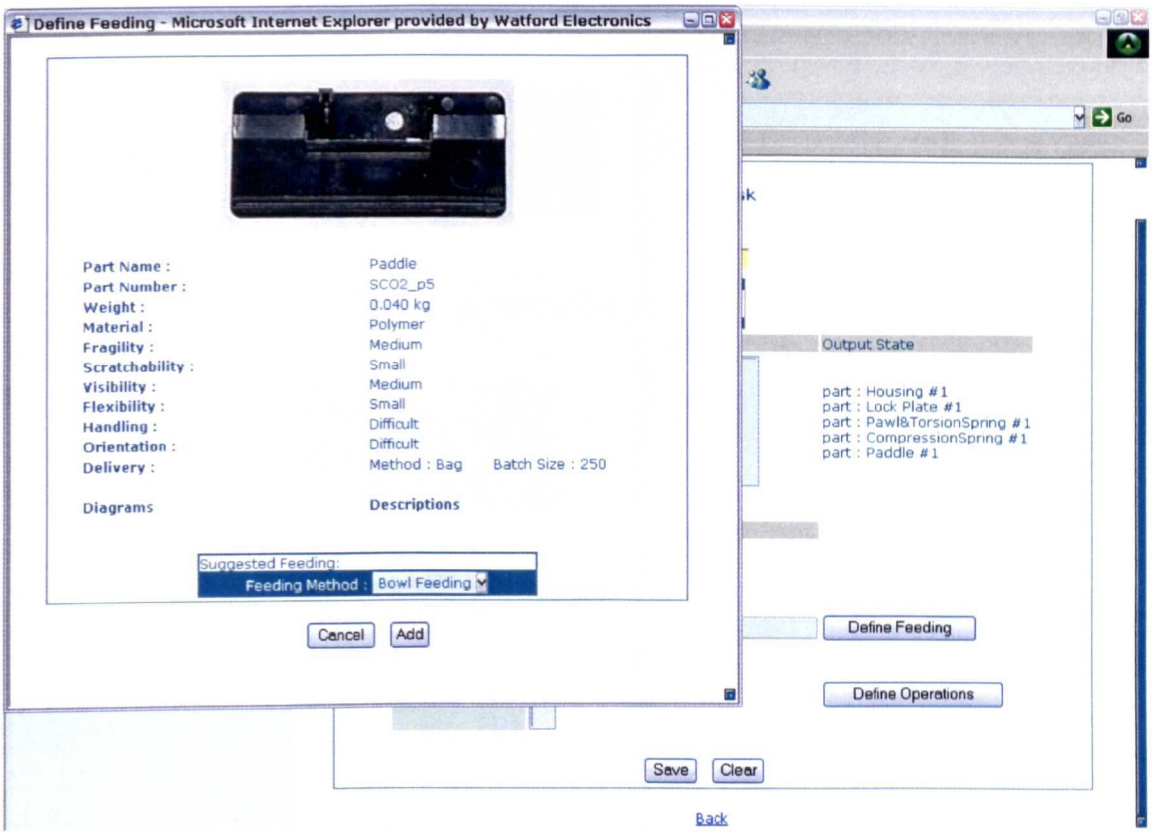


FIGURE 7-15: FEEDING SELECTION FOR PADDLE

Bowl feeding has been suggested by the inference engine as the part will arrive in bulk quantities (in bags) and there are no concerns about the scratchability of the part, which is a risk when using this method of feeding.

The assembly operation required for the task is automatically suggested through parsing of inference rules from the knowledge model based on the part liaison constraints and part data stated in the user requirements specification. This is illustrated in Figure 7-16.

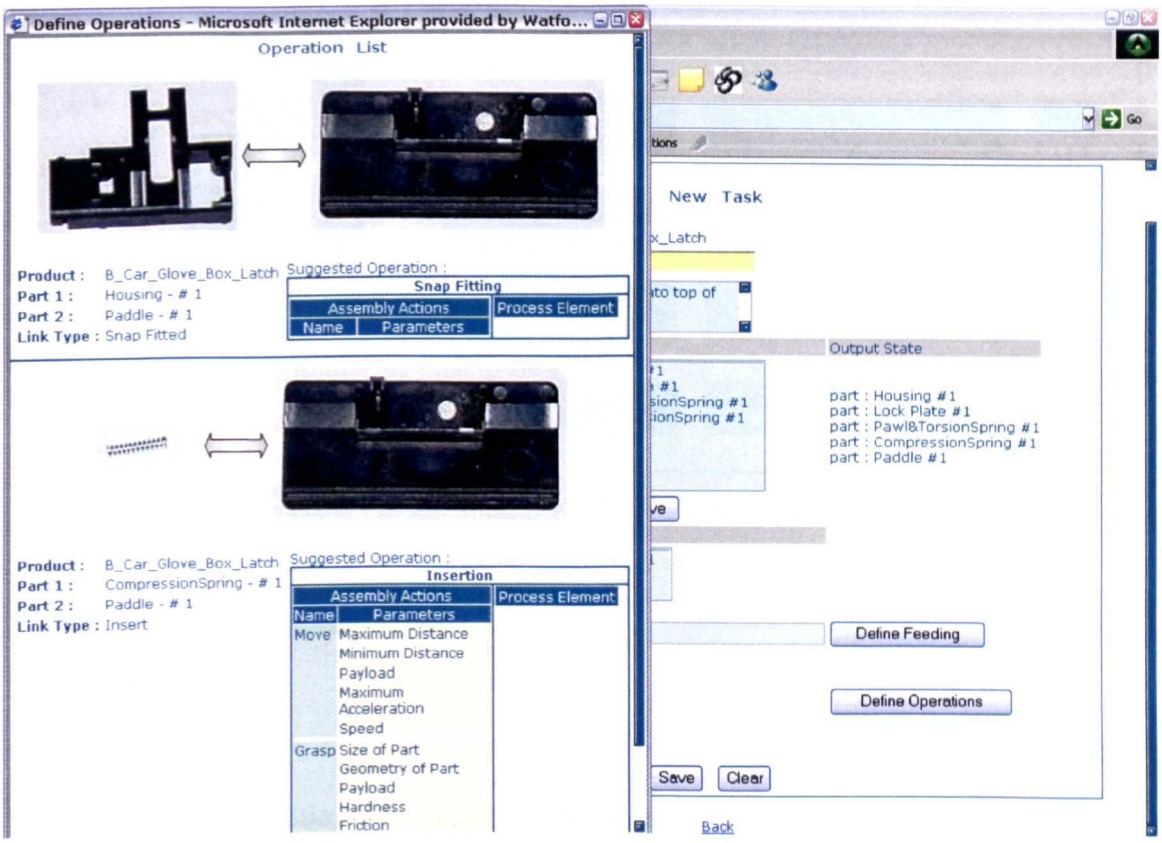


FIGURE 7-16: ASSEMBLY OPERATION SELECTION FOR INSERT PADDLE TASK

The insertion and snap fitting assembly operations are selected for the insert paddle task. These are specified simultaneously as the addition of one part to the product results in the establishment of two part liaisons (see Table 7-2). However, the similarity between the processes means that the same process elements will be needed to assemble the paddle.

In addition to the task specification, the system concept is specified through the "concept" option on the system requirements sub-environment menu (see Figure 7-17).

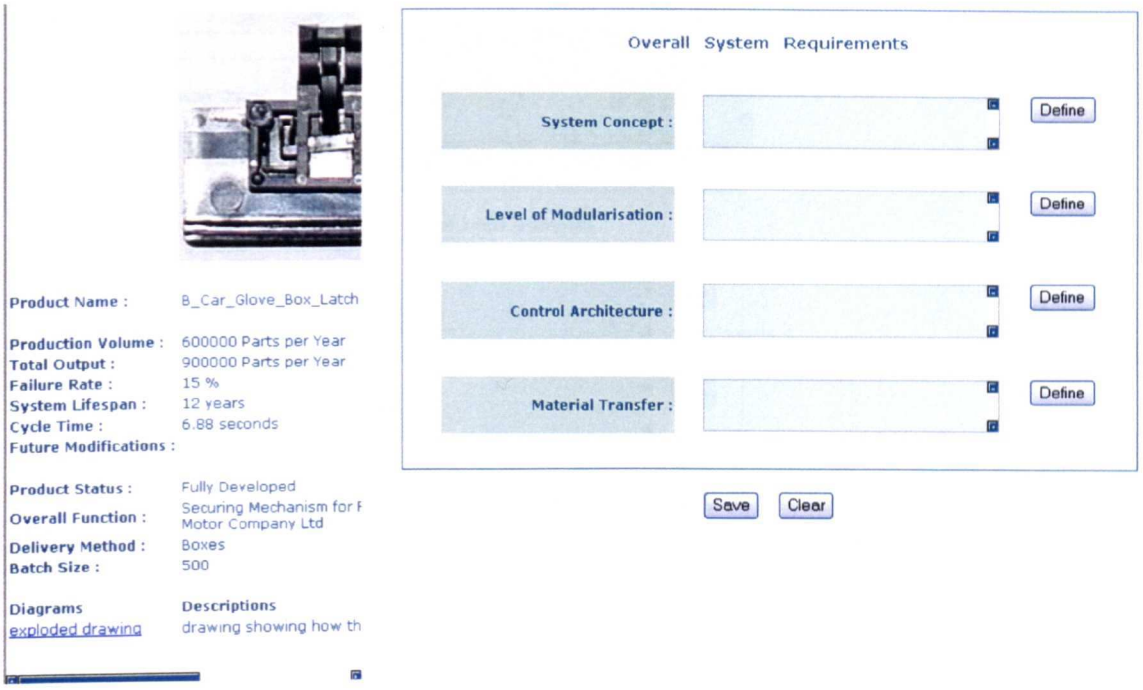


FIGURE 7-17: SYSTEM CONCEPT SCREEN FOR SCO2

System concept selection involves the selection of the overall concept, level of modularisation, control architecture and material transfer method. Product data is displayed on the left hand side of the page and overall system requirements are defined by pressing the "define" button next to the respective data items. A pop up screen appears where the recommended type is listed. This is based on the inference rules defined.

The system concept is the type of assembly that takes place on the line (see Figure 7-18). Conventional assembly is recommended in this case as the required cycle time is 6 seconds and the assembly tasks can be carried out within this time.

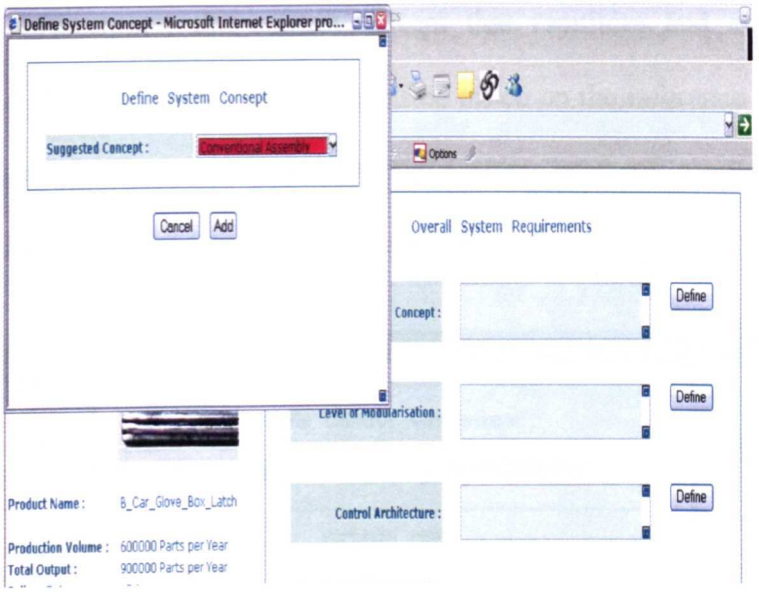


FIGURE 7-18: SYSTEM CONCEPT FOR SCO2

The next step is the definition of the level of modularity needed on the line. The recommended level is derived through automatic parsing of the inference rules. System Level Modularisation (see Figure 7-19) is selected as it is known that a second derivative of glove box latch is likely to be assembled on the line in the future, although the detailed design is not available at this point.

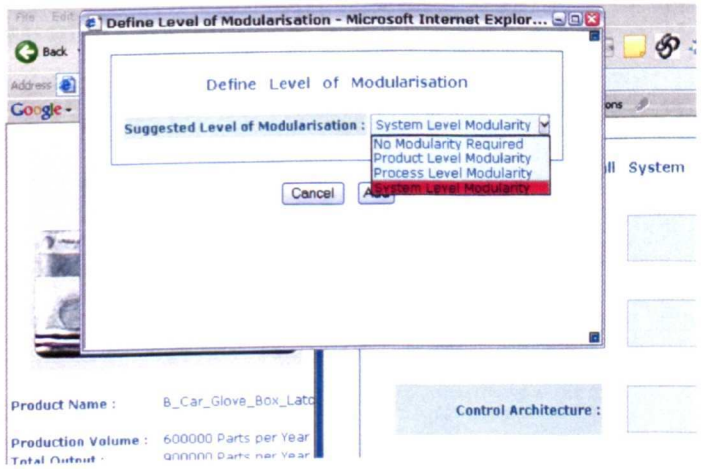


FIGURE 7-19: LEVEL OF MODULARISATION LEVEL FOR SCO2

Furthermore, the control system that monitors and controls the Reconfigurable Assembly System is also specified based on the inference rules. Distributed control is chosen as it complements the system level of modularisation, providing extra flexibility for future reconfiguration.

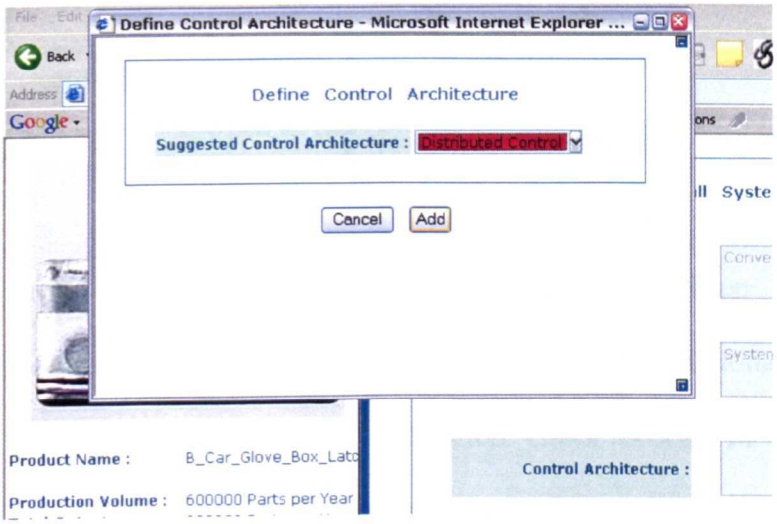


FIGURE 7-20: CONTROL ARCHITECTURE DEFINITION FOR SCO2

The materials handling method is recommended through automatic parsing of the inference rules. The material transfer requirements for the B car glove box latch assembly are illustrated in Figure 7-21.

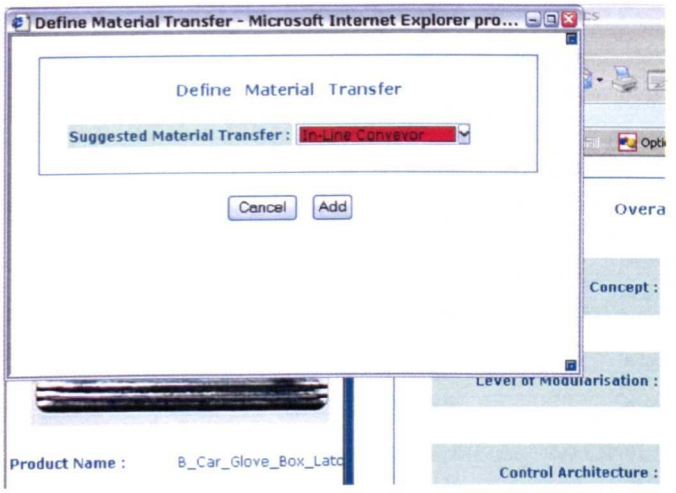


FIGURE 7-21: MATERIAL TRANSFER DEFINITION FOR SCO2

Once the functional system requirements have been formalised, the business requirements are evaluated to verify that the functional requirements meet the business requirements. A validation check is carried out using the inference knowledge to ensure that the system can be built within cost and time frameworks stated in the business requirements. These requirements are accepted, rejected or put aside for negotiation based on this validation. This is illustrated in Figure 7-22.

	Cost	Build Time
System Concept	£ 40000	40 days
Level of Modularisation	£ 30000	30 days
Control Architecture	£ 34000	10 days
Tasks :		
1. Insert Housing		
Insertion	£ 8000	15 days
Bowl Feeding	£ 2000	2 days
2. Insert Lock Plate		
Snap Fitting	£ 8000	15 days
Bowl Feeding	£ 2000	2 days
3. Insert Pawl		
Snap Fitting	£ 8000	15 days
Bowl Feeding	£ 2000	2 days
4. Insert Compression Spring		
Insertion	£ 8000	15 days
Bowl Feeding	£ 2000	2 days
5. Insert Paddle		
Snap Fitting	£ 8000	15 days
Bowl Feeding	£ 2000	2 days
Labour	£ 64000	120 days
Maintenance	£ 20000	120 days
	Total : £ 23800	**Total :** 165 days

Cost : 238000 - **Budget :** 250000 Accept Negotiate Decline

Build Time - 165 days - **Delivery Date :** 180 days Accept Negotiate Decline

FIGURE 7-22: SUMMARY OF COST AND BUILD TIME SYSTEM REQUIREMENTS FOR SCO2

A modular assembly system has been designed and implemented for assembly of this product and this is displayed in Figure 7-23.

FIGURE 7-23: SCO2 ASSEMBLY LINE

7.3.3 User Requirements Specification for Assembly System Reconfiguration

As the C Car Glove Box Latch has a similar product structure to the B Car latch, it is proposed that the same assembly system be used for the assembly of both products. Table 7-3 describes the differences to the individual parts for the C Car glove box latch assembly.

Part (From B Car)	**Modified Part (for C Car)**
Housing	Modified version to contain extra unit for key lock
Pawl & Torsion Spring	No modifications
Lockplate	Longer version to contain key lock
Compression Spring	No modifications
Paddle	Silver Paddle (Scratchable Surface)
Key Lock	New Part

TABLE 7-3: PARTS LIST FOR C CAR GLOVE BOX LATCH

Data for the new product is entered into the requirements engineering environment through the user requirements specification function. New business requirements and product details are described for the new product as per the user requirements for new Reconfigurable Assembly System. However, for this case, as the product is a modification from an already existing product (see Figure 7-24), parts are inherited from the original product (see Figure 7-26) and thereafter modified. The new part

entity inherits the properties of the original part and this data entity is then edited to include the part modifications as per Table 7-3 (see Figure 7-27).

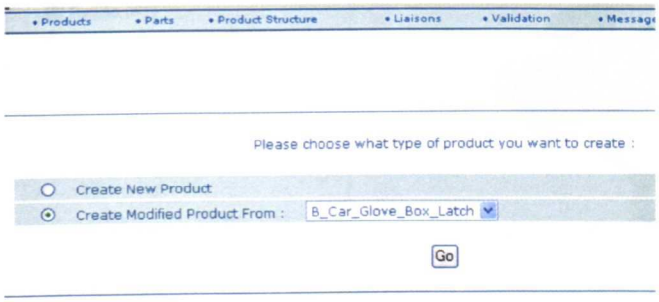


FIGURE 7-24: CREATING A NEW PRODUCT BASED ON EXISTING PRODUCT

The new product details are then specified as per Figure 7-25. This product is currently in design and this fact is brought to the attention of the system integrator, who can then shape the assembly system such that there is an element of flexibility to account for minor design changes that could take place between the current specification and the final design of the product.

FIGURE 7-25: PRODUCT DETAILS FOR C CAR GLOVE BOX LATCH

FIGURE 7-26: C CAR LATCH WITH INHERITED PRODUCT STRUCTURE

New liaison constraints are elicited to complete the user requirements data for the new product using the same methods as for the new Reconfigurable Assembly System specification process.

These requirements are then sent to the system requirements specification function for derivation of the system requirements.

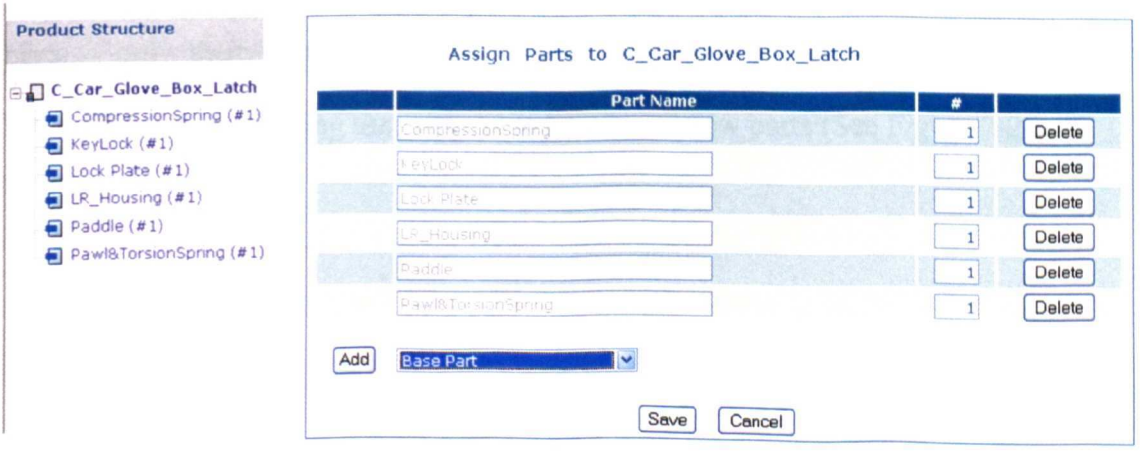


FIGURE 7-27: PART MODIFICATIONS FOR C CAR LATCH

7.3.4 System Requirements Specification for Assembly System Reconfiguration

The system requirements specification begins with the specification of assembly tasks by disassembly of the proposed product. Assembly operations and feeding methods are also derived in the same way as that for new assembly systems. Once the assembly tasks have been fully specified, a comparison is made between the task specification of the existing system and the task specification for the new product. Equivalent tasks are checked for equivalent operations to evaluate the type of reconfiguration that needs to take place.

The C Car glove box latch has the same pawl & torsion spring and compression spring as the B Car. Furthermore the fixtures on the housing in both cases are the same; hence the liaisons are also identical, making the assembly tasks for insertion of these items identical. This means that reconfiguration of the workstations that assemble these parts is not needed.

The modified housing, lockplate and paddle each need product level reconfiguration for their assembly workstations as the assembly processes have not changed for these parts – only their dimensions have changed and assembly of these parts can be achieved by reconfiguring the grippers to handle the new parts (See Figure 7-26).

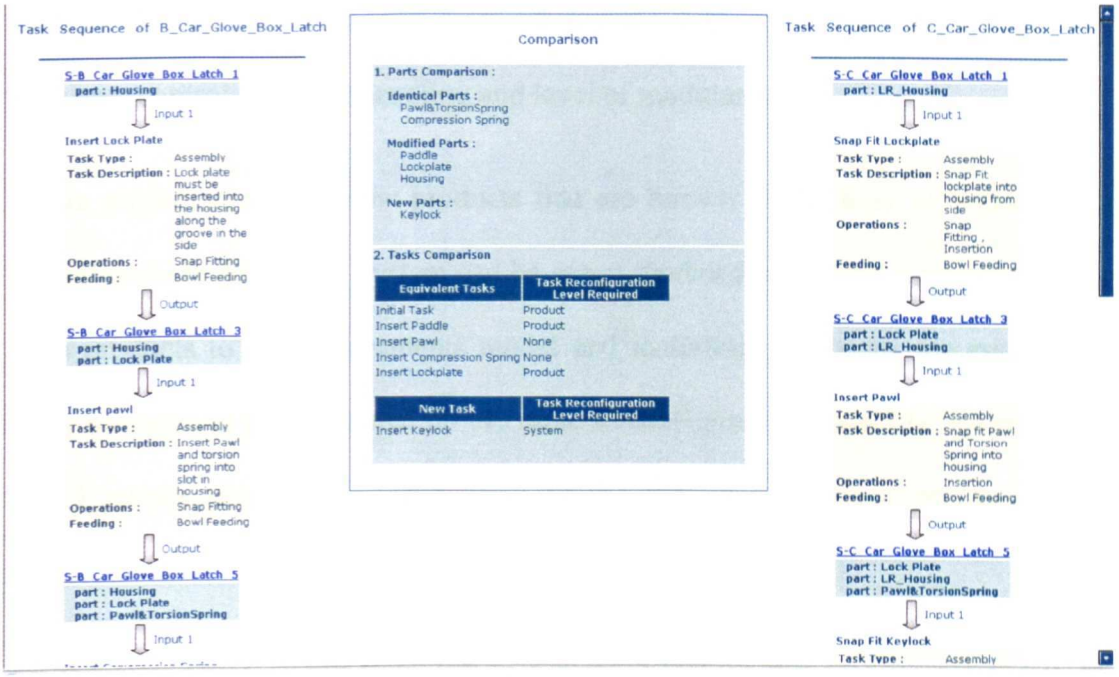


FIGURE 7-28: TASK AND PART COMPARISON FOR SCO2

However, the keylock is a completely new part and this part has to be assembled after the lockplate and before the compression spring. The inference engine concludes that a system level reconfiguration is needed to add a workstation on the assembly system to assemble the keylock.

7.4 DISCUSSION AND SYSTEM EVALUATION

Applications of requirements specification for Reconfigurable Assembly Systems to the Southco Glove box latch highlights the notion that the methods and tools created as part of this research serve their purpose.

In addition to the outcomes demonstrated through the case study, subassemblies can also be specified as complete subassemblies can be removed from the task specification by disassembly technique. Thereafter they are treated as individual

products in their own right as they can be assembled on separate assembly lines with their own control, material handling and level of modularisation.

Future modifications for new products that are known at the time of building the Reconfigurable Assembly System can be accommodated by adding requirements for extra products to the requirements model and including provisions for assembling many products to be assembled on the same Reconfigurable Assembly System. This would invoke physical reconfiguration of the system at product, process and/or system levels.

Assembly sequencing has been left as a manual task as this was considered to be outside the scope of the research because it was felt that there are many works in the field which aim to solve this problem. However, it is acknowledged that sequencing is a key activity in assembly system design and it should be possible to integrate other aspects of assembly system design into the already existing knowledge based system. This includes activities such as project planning, simulation and cost modelling. It is believed that these aspects are essential additions to make the research more attractive and practical to implement for companies.

In its present form, the environment includes only a very basic process model. This needs to be expanded to include a wider range of assembly processes with a greater number of assembly actions and process elements to make it a viable option for systems integrators.

The setup of the pilot environment has been designed so that new knowledge can always be added to the system. The author believes that the software will increase in effectiveness as more knowledge (facts and rules) is added to the knowledge base.

7.5 CHAPTER SUMMARY

A pilot web based environment has been created for requirements specification of one of a kind Reconfigurable Assembly Systems. The environment includes separate user interfaces for assembly system users and systems integrators. It facilitates the user requirements specification by the system user and system requirements specification by the systems integrator.

The pilot environment utilises knowledge at three levels of abstraction whereby task level knowledge is implemented through Java programs, inference knowledge through JESS and domain knowledge through a MySQL database.

The research outcomes are demonstrated through application to the specification of user requirements and system requirements for a Reconfigurable Assembly Systems to assemble two different car glove box latches. Requirements specification is performed according to the requirements specification model and methodology for a new assembly system and for system reconfiguration.

8 CONCLUSIONS AND FURTHER WORK

The research results open new opportunities for research in this field and further work needs to take place to make further advances in requirements specification for automated assembly. Industrial implementation is an important issue and steps need to be taken by industry to realise the true potential of the research. Original contributions to knowledge made by the research are stated below.

8.1 ORIGINAL CONTRIBUTIONS

The work has made advances in knowledge by addressing the following gaps:

- Limited Formalisation of Assembly Knowledge – before the research there was a distinct lack of formal methods of describing assembly operations. Much of the work in this area of manufacturing concentrated on line balancing and holistic design problems.

- Limited Application of Requirements Specification to Reconfigurable Assembly System Design – although requirements engineering has become an established field in software engineering, its application to engineering problems has been limited. Before this research, studies in this area concentrated on product design requirements where methods such as quality functional deployment have been used extensively. The application of requirements specification to assembly system design and furthermore to Reconfigurable Assembly Systems was limited.

- Limited Exploitation of Knowledge-Based Approaches in Industry – knowledge-based approaches, especially expert systems were prominent in the eighties but since then their application to engineering disciplines has been scarce. These approaches have hence been overseen as new assembly technologies, specifically Reconfigurable Assembly Systems have come to the fore.

The research presented in this thesis has contributed to filling the above knowledge gaps through the creation of:

- A Model and Methodology for Requirements Specification of Reconfigurable Assembly Systems
- A Knowledge Model for Supporting the Requirements Specification Model and Methodology for Reconfigurable Assembly Systems
- A Method of Assembly System Capability Modelling for Reconfigurable Assembly Systems

A new Assembly System Requirements Specification Methodology has been developed including the following key stages: define user requirements; analyse user requirements; define system requirements; requirements negotiation; and requirements verification.

User requirements have been described by classes relating to the product, its parts, liaisons between the parts and business constraints. The user requirements are then mapped to system requirements classes that refer to the assembly processes required to assemble the product within the business constraints. The system requirements

data model supports the decision making for the specification of new Reconfigurable Assembly Systems and reconfigurations to existing Reconfigurable Assembly Systems.

The requirements analysis process created here involves identifying user requirements that are either missing, ambiguous, conflicting or specified incorrectly. Any such requirements found are then sent for requirements negotiation between the system user and systems integrator. Requirements negotiation has been defined as a manual activity that involves the resolution of missing, conflicting, ambiguous or unacceptable requirements by the system user and systems integrator.

The requirements specification methodology includes requirements verification, which results in the acceptance or rejection of system requirements by the system user as the key activity. Any requirements that cannot be verified are negotiated by the system user and systems integrator.

A new knowledge model has been created whereby domain and inference knowledge has been structured to support the execution of tasks in user requirements specification, requirements analysis and system requirements specification.

"Pointer to domain" and "save data" inference rules have been proposed and a domain knowledge base has been developed to support user requirements specification. The domain knowledge stores user requirements under the following categories: project requirements; products; parts; and liaison characteristics.

Inference rules have been created that scan through domain knowledge to find missing requirements, incorrect data types and other conflicting requirements. Any

requirements that fit these criteria are sent for additional negotiation by the system user and systems integrator.

User requirements which are stored in the domain knowledge are mapped to system requirements through mapping inference rules. New domain knowledge is created that describes the project from a systems point of view, including project requirements and assembly task requirements. Hence domain and inference rules have been specified so that they can be easily updated as new assembly processes are developed or as new knowledge is discovered.

A new Assembly System Capability Model has been created that captures the inherent properties of assembly processes so that they can be mapped to assembly modules. Assembly operations are decomposed into assembly actions, which are grouped into process elements using a clustering algorithm. This provides the correct level of abstraction for mapping assembly modules to assembly processes.

The key development here is the discovery of common properties of assembly operations indicating the ability of assembly processes to be reconfigured at different levels of modularity. As a result several levels of reconfiguration have been defined:

- Product level reconfiguration means only a modification to the process parameters is made;
- Process level reconfiguration means that the assembly workstation is reconfigured to perform a different assembly process;

- System level reconfiguration implies a fundamental shift in the assembly system configuration, for example the addition or removal of assembly workstations or the conversation of a manual workstation into an automatic one.

The research outcomes are demonstrated through application to the specification of user requirements and system requirements for a Reconfigurable Assembly Systems to assemble two different car glove box latches. Requirements specification for the new assembly system and for system reconfiguration are presented.

8.2 FURTHER WORK

The research presented here opens new opportunities for further advancing technological development and industrial practice. There are three main areas where these opportunities arise.

Application of the Requirements Specification Model to Other Scenarios

The requirements specification model and methodology covers the process from capturing the customers' requirements to extracting system requirements. These form the basis for conceptual design and integrating this model with conceptual design methods will bring about great benefit to industry. Furthermore, the model can be adapted for application to requirements specification for other assembly systems such as flexible or fixed assembly systems.

Cost modelling and project planning were omitted from the project due to limitations on data and time resources. However there is sufficient flexibility in the framework

to include these so that systems integrators can accurately predict their costs and timelines for assembly system design and deployment projects. Moreover the creation of plug-ins to accounting systems will provide opportunities for creating more accurate tenders and a vital medium for promoting automated assembly to accountants that make financial decisions in small and medium sized enterprises (SMEs).

Customisation of the Assembly System Capability Model

The assembly system capability model has established a format and structure for choosing assembly operations based on operation functionality. This model is based on process knowledge extracted from case studies investigated during the course of the project. This framework can be exploited by other aspects of manufacturing industry with formal definitions and technical specifications for machining and testing operations. The assembly system capability model has set a precedence for formalising future research on manufacturing processes.

Commercial Web Portal for Specification and Delivery of Reconfigurable Assembly System Solutions

Knowledge-enriched requirements specification for reconfigurable assembly systems demonstrates how knowledge can be used to specify reconfigurable assembly systems. This concept can be extended to create an electronic market where different system users and system integrators can interact with each other. System users can specify their requirements and invite system integrators to submit tenders for satisfying those requirements. The system integrators, in turn can use the portal for

interacting with their equipment suppliers and ordering components from them online. More standardised paths for communication between customers and suppliers of assembly systems can be realised and through such initiatives, the outcomes of this thesis can be used for the sustainable development of European assembly.

8.3 CONCLUDING REMARKS

The accurate specification of reconfigurable assembly systems is an important issue to integrate into design, which in turn, will strengthen manufacturing industry to reach new levels of competitiveness based on reconfigurable automation. This work adds a new dimension to the movement towards Reconfigurable Assembly Systems.

A new method and tools for requirements specification of reconfigurable assembly systems have been created. This includes:

- a requirements specification model and methodology
- a knowledge model to support the requirements specification methodology
- a process capability model that captures the ability of different assembly operations to be reconfigured

The applicability of the research results has been demonstrated by the development of a prototype environment and its application to an industrial case study.

REFERENCES

MR Abdi; AW Labib (2003): "A Design Strategy for Reconfigurable Manufacturing Systems Using Analytical Hierarchical Process: A Case Study," International Journal of Production Research, 41:10, pp2273-2299

T Arai; Y Aiyama; Y Maeda; M Sugi (2001): "Holonic Robot System: A Flexible Assembly System With High Reconfigurability", Proceedings of The 2001 International Conference on Robotics and Automation, Seoul, Korea, May 21st-26th

C Archibald; E Petriu (1993): "Computational Paradigm for Creating and Executing Sensor-Based Robot Skills," Proceedings of the 24th International Symposium on Industrial Robots (ISIR), Tokyo, Japan, November.

MC Becker; F Zirpoli (2003): "Organising New Product Development: Knowledge Hollowing-Out and Knowledge Integration – The FIAT Auto Case," International Journal of Operations & Production Management, 23:9, pp1033-1061

R Belecheanu; KS Pawar; RJ Barson; B Bredehorst; F Weber (2003): "The Application of Case Based Reasoning to Decision Support in New Product Development," Integrated Manufacturing Systems, 14:1, pp36-45

J Bollinger (1998): "Visionary Manufacturing Challenges for 2020," National Research Council Publication, Washington DC (USA)

L Bongaerts; H Van Brussel; P Valckenaers (2003): "Generic Concepts For Holonic Manufacturing Control Department of Mechanical Engineering," Katholieke

Universiteit Leuven, Belguim, Http://www.mech.kuleuven.ac.be/pma/welcome.html, (August 2003)

I K Bray (2002): "An Introduction to Requirements Engineering," Pearson Education Ltd, Harlow, Essex (UK) ISBN 0201 767929

M Broman; S Eskilander (2000): "A Tool for Assembly System Design," TuDelft Assembly Automation Workshop, 11th-12th May, Delft University of Technology, The Netherlands

JL Burbridge (1989): "Production Flow Analysis for Planning Group Technology," Clarendon Press, Oxford, (UK)

JJ Carr (2000): "Requirements Engineering and Management," The TQM Magazine, 12:6, pp400-407

S Chakraborty; J Wolter (1994): "A Hierarchical Approach to Assembly Planning," IEEE International Conference on Robotics and Automation, San Diego, May 1994, pp58-63

FTS Chan; B Jian (1999): "Simulation Aided Design of Production and Assembly Cells in an Automotive Company," Integrated Manufacturing System, 10:5, pp276-283

CW Chan; Y Peng; LL Chen (2003): "Knowledge Acquisition and Ontology Modelling of Construction of a Control and Monitoring Expert System, International Journal of Systems Science," 33:6, pp485-503

HL Chau; EJ Derrick; HC Shen; RK Wong (1995): "A New Approach for the Specification of Assembly Systems," IEEE 0-8186-6995-0/95 pp9-14

I-M Chen (2001): "Rapid Response Manufacturing through a Rapidly Reconfigurable Robotic Cell," Robotics and Computer Integrated Manufacturing, 17, pp199-213

K Cheng; PY Pan; DK Harrison (2000): "The Internet as a Tool with Application to Agile Manufacturing: A Web-Based Engineering Approach and its Implementation Issues," International Journal of Production Research, 38:12, pp2743-2759

SE Chick; TL Olsen; K Sethuram; KE Stecke; CC White III (2000): "A Descriptive Multi-Attribute Model for Reconfigurable Machining System Selection Examining Buyer-Supplier Relationships," International Journal Of Agile Manufacturing Systems, 2:1, pp33-48

C Chung; Q Peng (2004): "The Selection of Tools and Machines an Web-Based Manufacturing Environments," International Journal Of Machine Tools and Manufacture, 44, pp317-326

R Cowan (2001): "Expert Systems: Aspects and Limitations to the Codifiability of Knowledge," Research Policy, 30, pp1355-1372

MJ Darlington; SJ Culley (2002): "Current Research in the Engineering Design Requirement," Proceedings of the Institute of Mechanical Engineers, Journal of Engineering Manufacture, 216:B, pp375-388

J Debenham (1998): "Knowledge Engineering – Unifying Knowledge Base and Database Design," Springer Press, Heidelberg, Germany

A Delchambre (1996): "CAD Method for Industrial Assembly: Concurrent Design of Products, Equipment and Control Systems," Wiley and Sons Inc, Chichester, West Sussex (UK)

V Dignum, (1999): "Knowledge Management for Requirement Engineering, Proceeding of the Twelfth Workshop on Knowledge Acquisition, Modelling and Management," KAW'99 Voyager Inn, Banff, Alberta, Canada

G Dini (2002): "Assembly-Net Assembly System Taxonomy," Assembly-Net Internal Publication, www.Assembly-Net.org

A Dugenske; A Fraser; T Nguyen; R Viotus (2000): "The National Electronics Manufacturing Initiative (NEMI) Plug & Play Factory Project," International Journal of Computer Integrated Manufacturing, 13:3, pp225-244

SM Easterbrook (1991): "Elicitation of Requirements from Multiple Perspectives," PhD Thesis, Imperial College of Science, Technology and Medicine, University of London, London, UK

NF Edmondson; AH Redford (2002): "Generic Flexible Assembly System Design," Assembly Automation, 22:2, pp139-152

K Feldmann; H Rottbauer (2000): "Electronically Networked Assembly Systems for Global Manufacturing," Journal of Materials Processing Technology, 107:1, pp319-329

G Fliedl; C Kop; HC Mayr; W Mayerthaler; C Winkler (2000): "Linguistically Based Requirements Engineering – The NIBA Project," Data & Knowledge Engineering, 35, pp111-120

N Gardan; Y Gardan (2003): "An Application of Knowledge Based Modelling Using Scripts," Expert Systems with Applications, 25, pp555-568

C Grabowik; R Knosala (2003): "The Method of Knowledge Representation for a CAPP System," Journal of Materials Processing Technology, 133, pp90-98

J Heilala; P Voho (2001): "Modular Reconfigurable Flexible Final Assembly Systems," Assembly Automation, 21:1, pp20-28

H Hirani; S Ratchev (2002a): "Knowledge Specification for Requirements Engineering of Reconfigurable Assembly Systems," Proceedings of the 18th International Conference on CAD, CAM, Robotics and Factories of the Future, Porto, Portugal, June 2002

H Hirani; S Ratchev (2002b): "Definitions and Measures for Requirements Engineering of Reconfigurable Assembly Systems," Proceedings of the 31st International Symposium on Robotics, Stockholm, Sweden, October 2002

H Hirani; S Ratchev (2003): "Operations Capability Model for Requirements Engineering of Reconfigurable Precision Assembly Systems," Proceedings of the 1st International Precision Assembly Seminar, Bad Hofgastein, Austria, March 2003

H Hirani: S Ratchev: N Lohse; George Valtchanov (2004): "Web-Based Specification of Reconfigurable Precision Assembly Systems - Industrial Scenarios

and Use Cases," Proceedings of the 2nd International Precision Assembly Seminar, Bad Hofgastein, Austria, February 2004

H Hirani: S Ratchev: N Lohse; George Valtchanov (2004): "Web-Based Specification of Reconfigurable Precision Assembly Systems - Industrial Scenarios and Use Cases," Proceedings of the Intelligent Manufacturing Systems International Forum, Lake Como, Italy, May 2004

JK Ho; PG Ranky (1997): "Object Oriented Modelling and Design of Reconfigurable Conveyors in Flexible Assembly Systems," International Journal of Computer Integrated Manufacturing, 10:5, pp360-379

R Hollis; A Quaid (1995): "An Architecture For Agile Assembly," Proceedings of the American Society of Precision Engineering, 10th Annual Meeting, Austin, TX (USA)

SH Huang; X Hao; M Benjamin (2001): "Automated Knowledge Acquisition for Design and Manufacturing: The Case of Micromachined Atomizer," 12, pp377-391

CC Huang; A Kusiak (1998): "Modularity in Design of Products and Systems," IEEE Transactions on Systems, Man, and Cybernetics, Part A, 28:1, 66-77

BL Huff; CR Edwards (1999): "Layered Supervisory Architecture for Reconfigurable Automation," Production Planning & Control, 10:7, pp659-670

M Jackson (1997): "The Meaning of Requirements," Annals of Software Engineering 3, pp5-21

R B Jackson; DW Embley; SN Woodfield (1995): "Developing Formal Object Oriented Requirements Specifications: A Model, Tool and Technique," Information Systems, 20:4, pp273-289

J Jiao; MM Tseng (1999): "A Requirement Management Database System for Product Definition," Integrated Manufacturing Systems, 10:3, pp146-153

J Jiao; MM Tseng (2000): "Fundamentals of Product Family Architecture," Integrated Manufacturing Systems, 11:7, pp469-483

R Johansson (2002): "Implementation of Flexible Automatic Assembly Systems in Small Companies," Doctoral Thesis, Department of Production Engineering, Royal Institute of Technology, Stockholm, Sweden

S Jordan (1997): "Modular Assembly: A Process Not an Engineering Technique," Assembly Automation, 17:4, pp282-286

Y Koren; AG Ulsoy (1997): "Reconfigurable Manufacturing Systems," Engineering Research Centre for Reconfigurable Manufacturing Systems (ERC/RMS) Report #1, The University of Michigam, Ann Arbor.

G Kotonya; I Sommerville (1998): "Requirements Engineering: Processes and Techniques," John Wiley & Sons Ltd., Chichester, West Sussex (UK), ISBN 0471972088

A Kumar; SH Jacobson; EC Sewell (2000): "Computational Analysis of a Flexible Assembly System Design Problem," European Journal of Operational Research, 123, pp453-472

A Kusiak (1999): "Engineering Design: Products, Processes and Systems," Academic Press, London (UK), ISBN 0124301422

W. Lam (1997): "Achieving Requirements Reuse: A Domain Specific Approach from Avionics," Journal of Systems Software, 38, pp197-209.

SH Liao (2003): "Knowledge Management Technologies and Applications – Literature Review from 1995-2002," Expert Systems with Applications, 25, pp155-164

M Lindvall; I Rus; SS Sinha (2003): "Software Systems Support for Knowledge Management," Journal of Knowledge Management, 7:3, pp137-150

N Lohse; H Hirani; S Ratchev (2003): "Task Based Assembly Planning and Configuration of Assembly Workstations," Proceedings of the International Symposium on Assembly and Task Planning, Besancon, France, July 2003

B Lotter (1989): "Manufacturing Assembly Handbook," Addison Wesley, London (UK)

LA Macaulay (1999): "Seven-Layer Model of the Role of the Facilitator in Requirements Engineering," Requirements Engineering, 4, pp38-59

M Martensson (2000): "A Critical Review of Knowledge Management as a Management Tool," Journal of Knowledge Management, 4:3 pp 204-216

P Martensson (2000): "Manufacturing Subsystem Design Decisions," Proceedings of the 33rd International Seminar on Manufacturing Systems, 5th-7th June, Stockholm, Sweden, pp60-65

R Mason-Jones; D Berry; MM Naim, (1998): "A Systems Engineering Approach to Manufacturing Systems Analysis," Integrated Manufacturing Systems, 9:6, pp350-365

R McAdam; S McCreedy (1999): "A Critical Review of Knowledge Management Models," The Learning Organisation, 6:3, pp.91-100

AS McCampbell; LM Clare; SH Gitters (1999): "Knowledge Management: The New Challenge for the 21st Century," Journal of Knowledge Management, 3:3, pp172-179

MG Mehrabi; AG Ulsoy; Y Koren; P Heytler (2002): "Trends and Perspectives in Flexible and Reconfigurable Manufacturing Systems," Journal of Intelligent Manufacturing, 13, pp135-146

K Metaxiotis; J Psarras (2003): "Expert Systems in Business: Applications and Further Directions for the Operations Researcher," Industrial Management and Data Systems, 103:5, pp361-368

RP Monfared; RH Weston (1997): "The Re-Engineering and Reconfiguration of Manufacturing Cell Control Systems and Reuse of Their Components," Proceedings of The Institution of Mechanical Engineers Conference, 211:B, pp495-508

PF Muir; AA Rizzi; Jay Gowdy (1997): "Architectures, Networks and Intelligent Systems for Manufacturing Integration," Proceedings of The SPIE, Volume 3203, pp74-80

N Nakasa; T Yamada; M Matsui (2002): "A Management Design Approach to a Simple Flexible Assembly System," International Journal of Production Economics, 76, pp281-292

MM Nkasu; KH Leung (1995): "Computer-Integrated Manufacturing Assembly System Design," Integrated Manufacturing Systems, 6:6, pp4-14

M Onori; J Barata; J Lastra; M Tichem (2003): "European Precision Assembly Roadmap 2012," Assembly-Net Internal Publication, www.Assembly-Net.org

OUP (1996): "Concise Oxford Dictionary" (10th Ed), OUP, Oxford, UK

HK Rampersad (1993): "Integrated and Simultaneous Design for Robotic Assembly," John Wiley and Sons, Chichester, West Sussex (UK)

P G Ranky (1998): "Some Aspects of Real Time Production Control in Distributed Flexible Assembly Systems," Assembly Automation, 18:1, Pp57-67

S Ratchev; J Shiau; G Valtchanov (2000): "Distributed Product and Facility Prototyping in Extended Manufacturing Enterprises," International Journal of Production Research, 38:17, pp 4495 – 4506

S Ratchev; H Hirani (2001a): "Requirement Engineering for Re-configurable Assembly Cells," Proceedings of the 34th International Seminar on Manufacturing Systems, Athens, Greece, May 2001

S Ratchev; H Hirani (2001b): "Concurrent Requirement Specification for Conceptual Design of Modular Assembly Cells," Proceedings of the International Symposium on Assembly and Task Planning, Fukuoka, Japan, May 2001

S Ratchev; H Hirani; N Lohse (2004): "Knowledge Model for the Configuration of Modular Assembly Worksations – Specification and Distributed Conceptual Design," Proceedings of the 35th International Symposium on Robotics, Paris, France, March 2004

A Rizzi; J Gowdy; R Hollis (1997), "Agile Assembly Architecture: An Agent Based Approach to Modular Precision Assembly Systems," Proceedings of the 1997 International Conference on Robotics and Automation

M Robertson; G O'Mally (2000): "Knowledge Management Practices within a Knowledge Intensive Firm: The Significance of the People Management Dimension," Journal of European Industrial Training, 24/2/3/4/ 2000 pp.241-253

H Scarborough (2000): "Knowledge Management – A Literature Review," Internal Document, University of Leicester, Leicester (UK)

G. Schreiber; B. Wielinga; and J. Breuker (1993): "KADS: A Principled Approach to Knowledge-Based System Development," Academic Press, Sidcup, Kent (UK)

G Schreiber; H Akkermans; A Anjewierden, R de Hoog, N Shadbolt; W Vande Velde, B Weilinga (2000): "Knowledge Engineering And Management – The CommonKADS Methodology," The MIT Press, Cambridge, Massachusetts (USA)

AP Sinha; D Popken (1996): "Completeness and Consistency Checking of System Requirements: An Expert Agent Approach," Expert Systems with Applications, 2:3 pp263-276

P C Stadzisz; J M Henrioud (1998): "An Integrated Approach for the Design of Multi-Product Assembly Systems," Computers in Industry, 36:1, pp21-29

R Stevens; J Martin (1995): "What is Requirements Management?" Proceedings of the 5th Annual International Symposium of the NCOSE, Vol 2, pp13-18

R Stevens; K Jackson; P Brook; S Arnold (1998): "Systems Engineering: Coping with Complexity," Prentice Hall PTR, ISBN 0130950858

M Tichem; T Storm; J Vos (1999): "How to Achieve a Breakthrough in Industrialisation of Flexible Assembly Automation," Proceedings of the Ninth International FAIM Conference, 23rd-25th June 1999, Tilberg, The Netherlands, pp 327-336

P Valckenaers (1993): "Flexibility for Integrated Production Automation," Doctoral Thesis; Katholieke Universiteit Leuven, Departement Werktuigkunde, Belgium.

JAWM Vos (2000): "Design of a Flexible Industrial Assembly System," PhD Thesis, Laboratory for Production Engineering and Industrial Organisation, Delft University of Technology, Delft, The Netherlands

KM Wiig (1997): "Knowledge Management: An Introduction and Perspective," Journal of Knowledge Management, 1:1, pp6-14

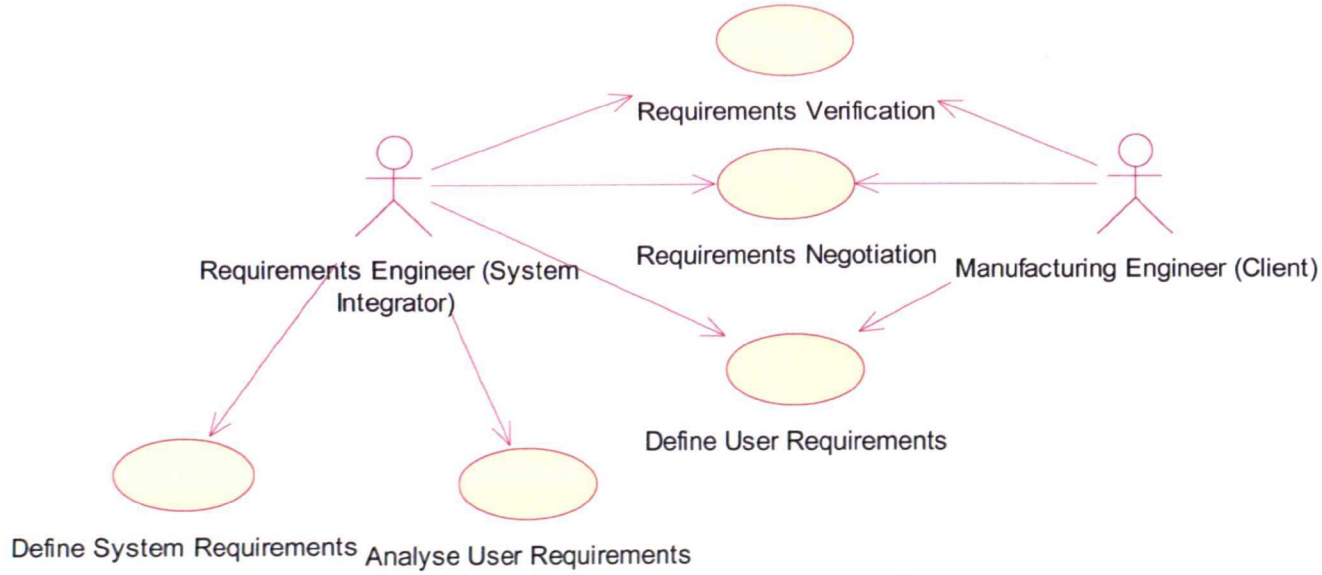
N Ye; D Urzi (1999): "Heuristic Rules and Strategies of Assembly Planning: Experiment and Implications in the Design of Assembly Decision Support System," International Journal of Production Research; 34:8

www.commonKADS.uva.nl, 5th March 2001

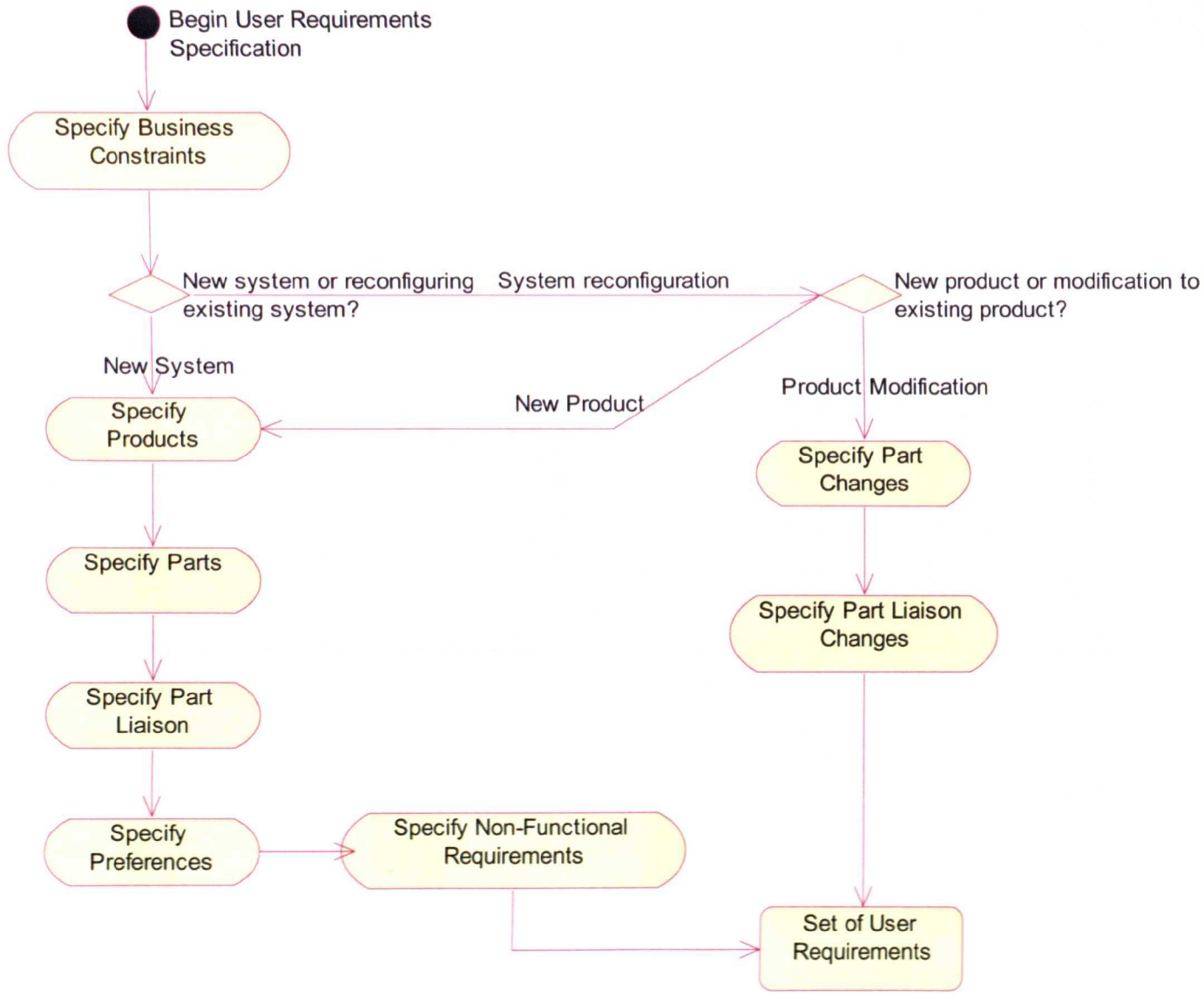
www.assembly-net.org, Assembly-Net Consortium, 12th January 2004

APPENDIX A

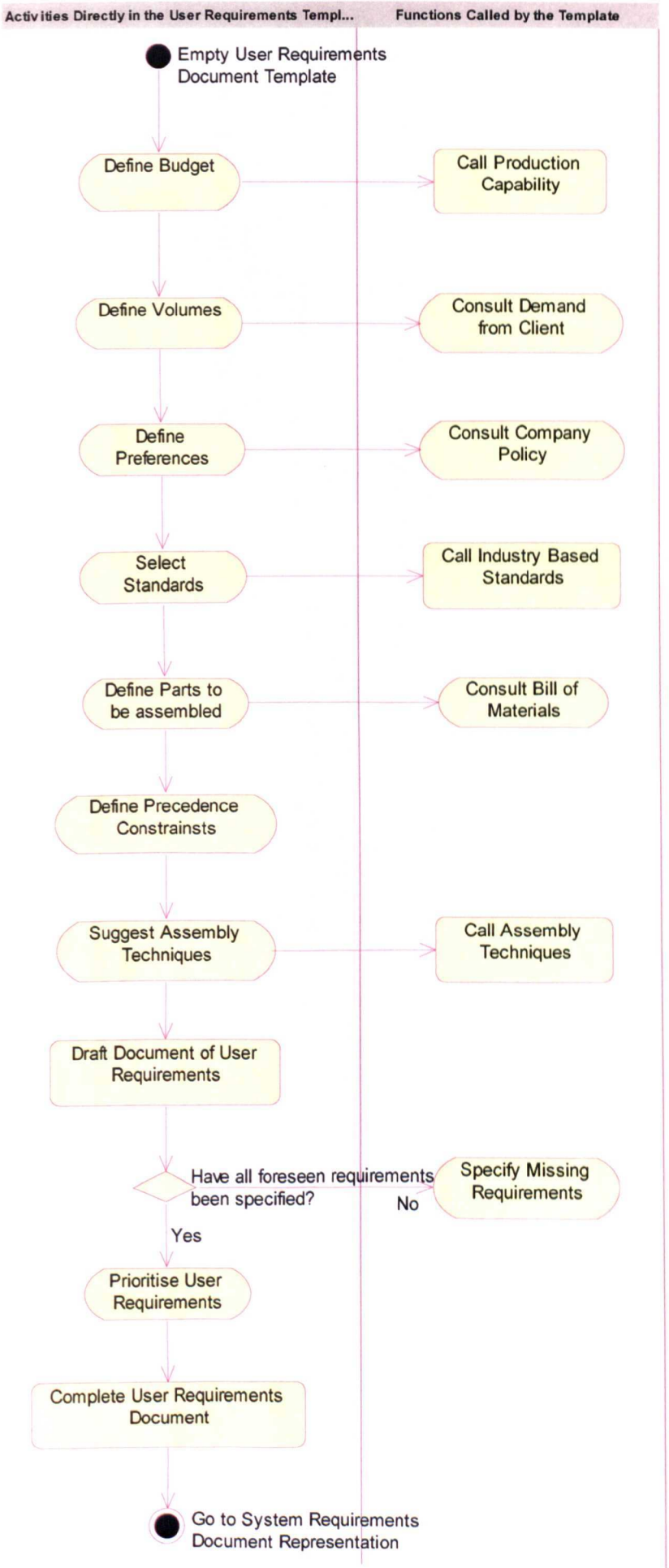
USE CASE DESCRIPTIONS



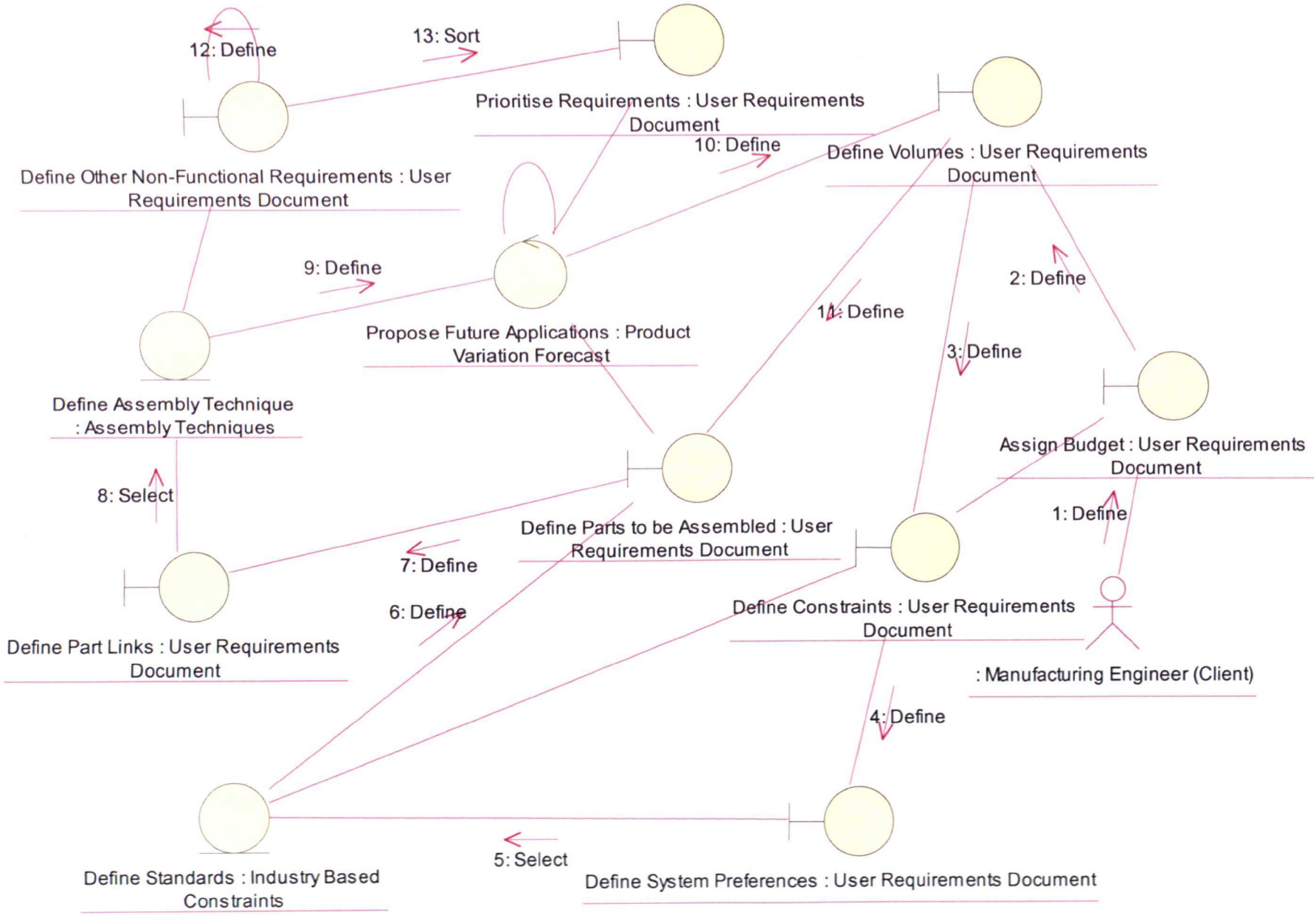
Summary of Requirements Engineering Use Cases



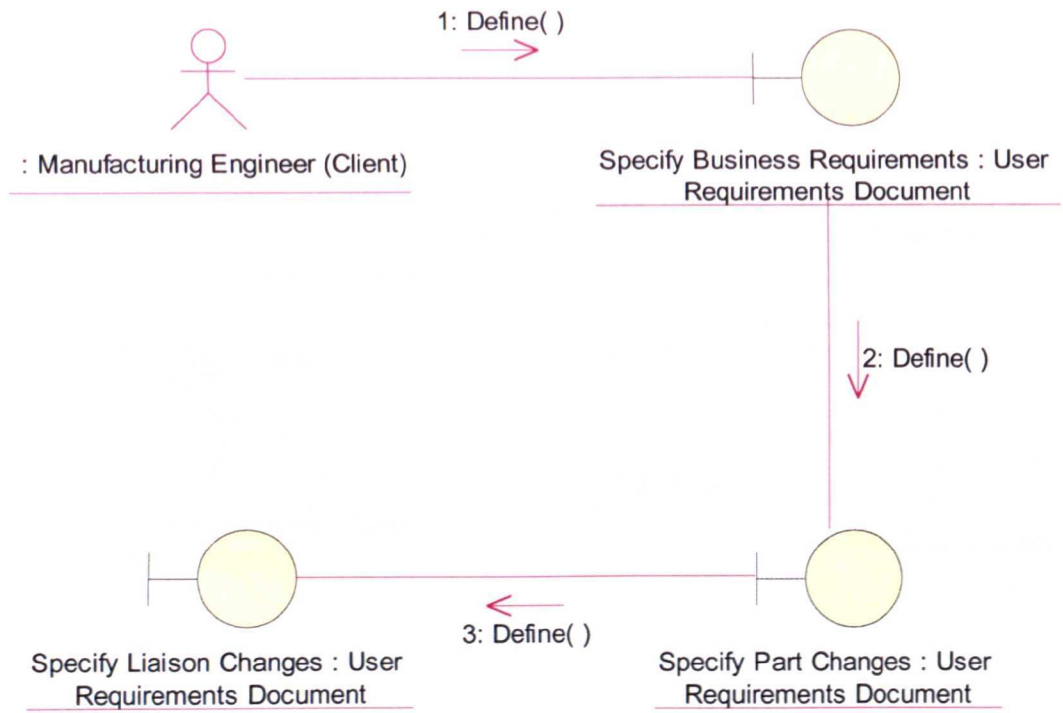
Decision Tree for User Requirements Specification



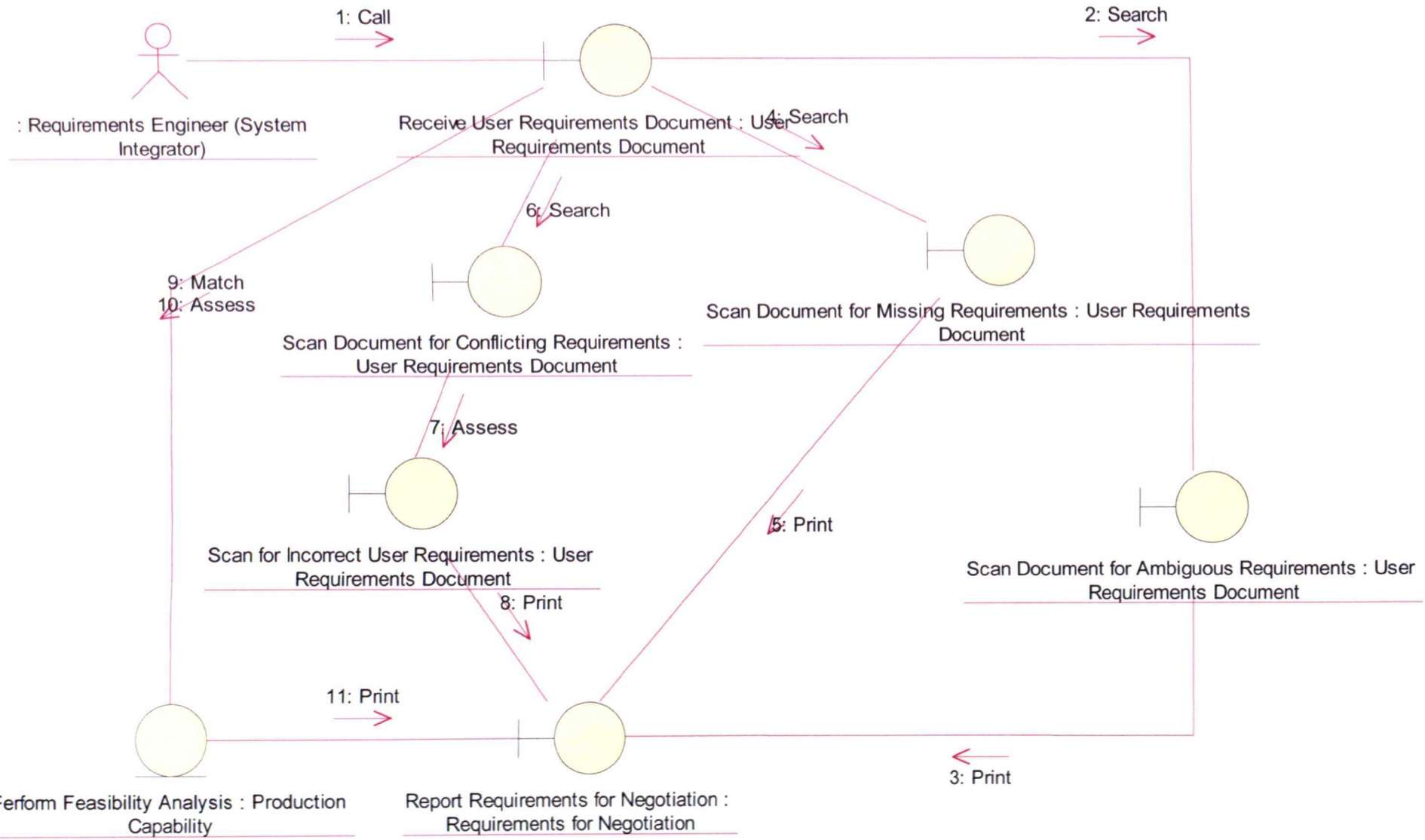
Activity Set for User Requirements Document



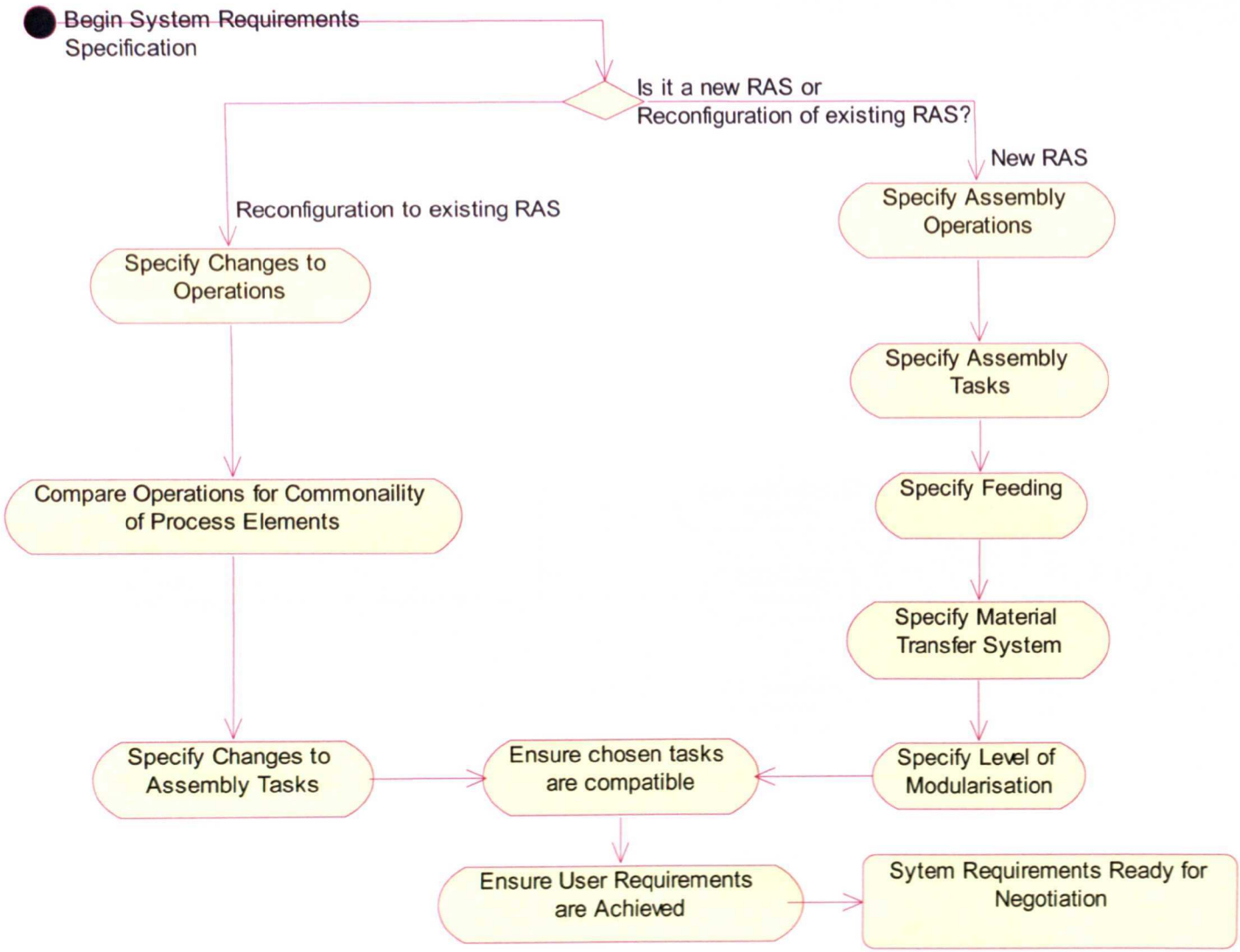
Collaboration Diagram Collaboration Diagram for User Requirements Specification



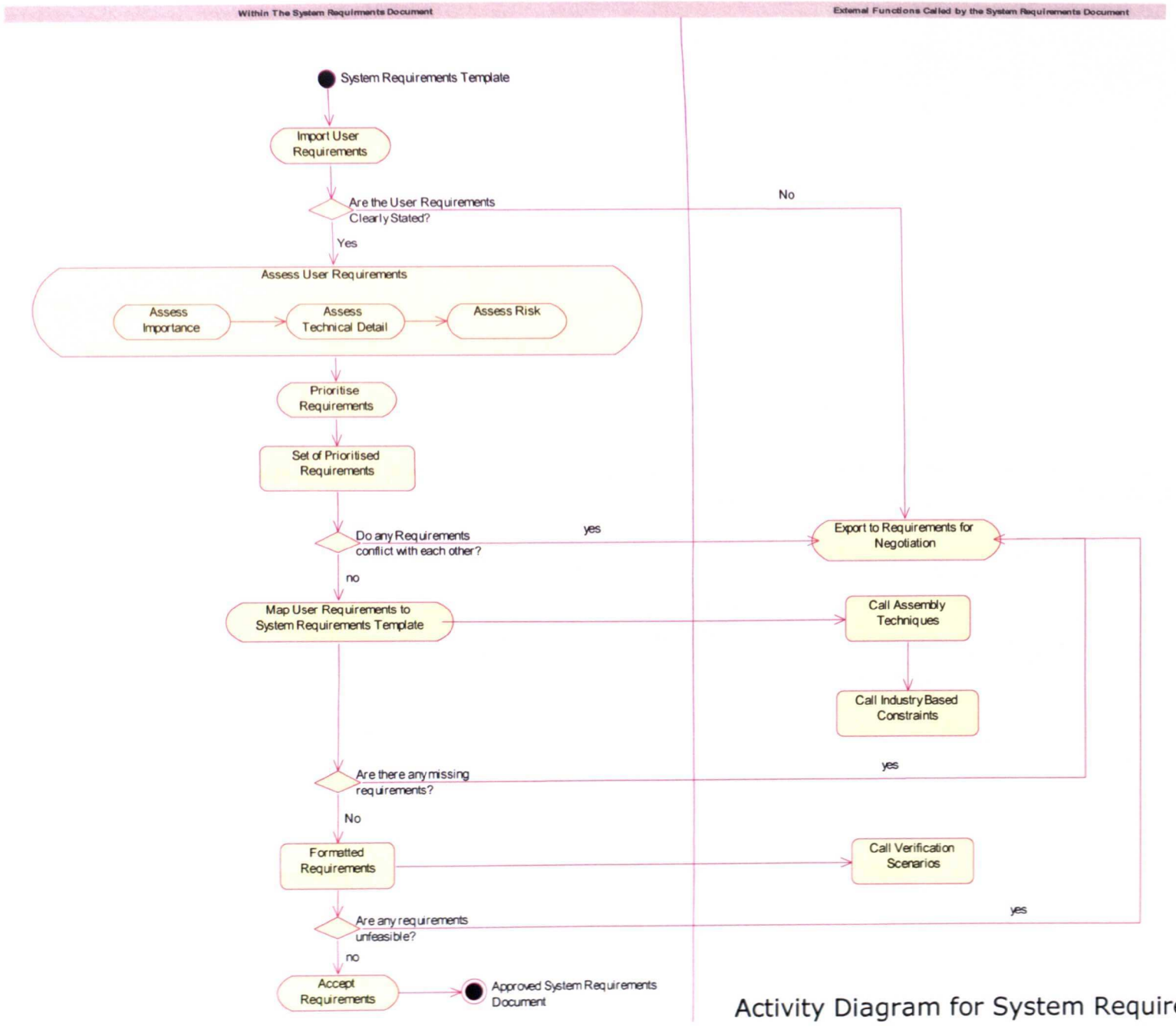
Collaboration Diagram for User Requirements Specification (System Reconfiguration)



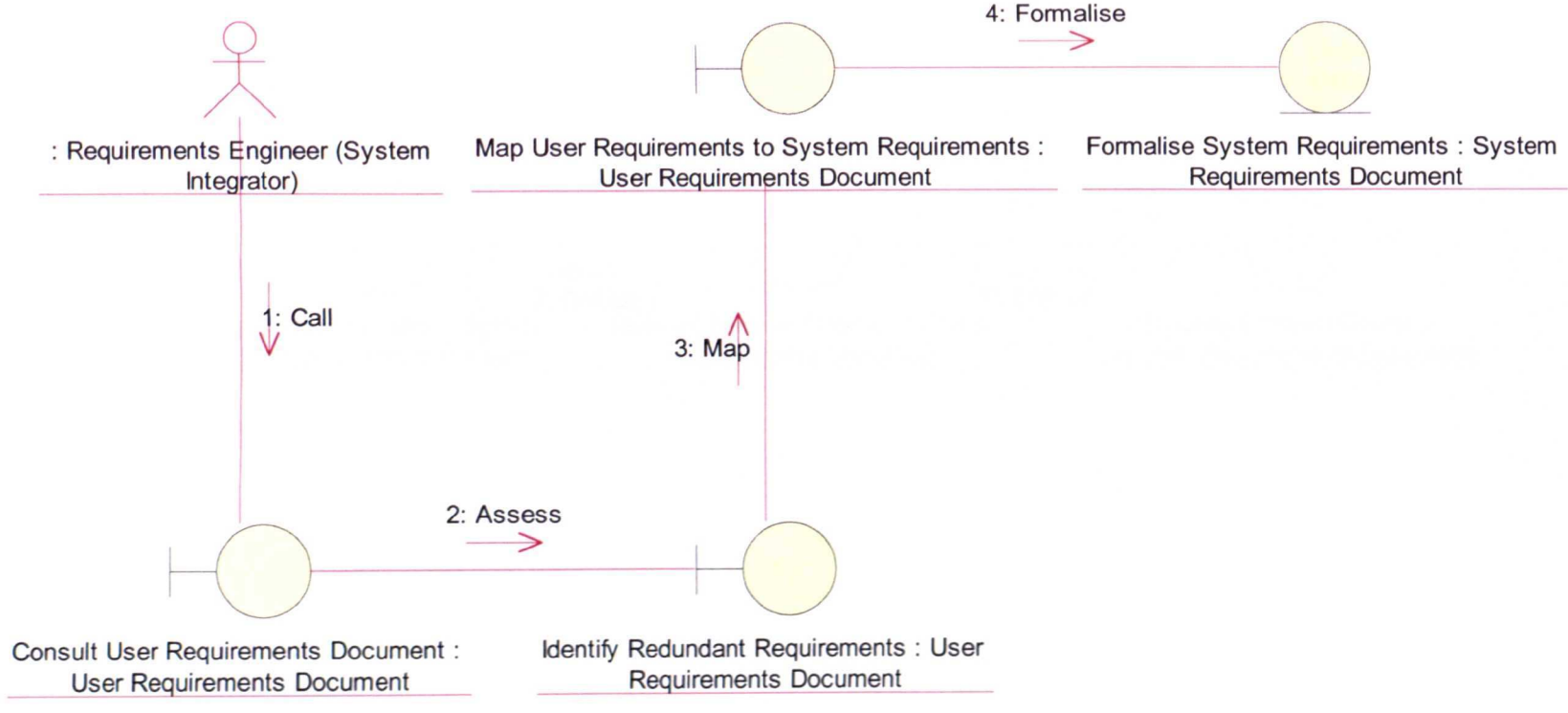
Collaboration Diagram for Requirements Analysis



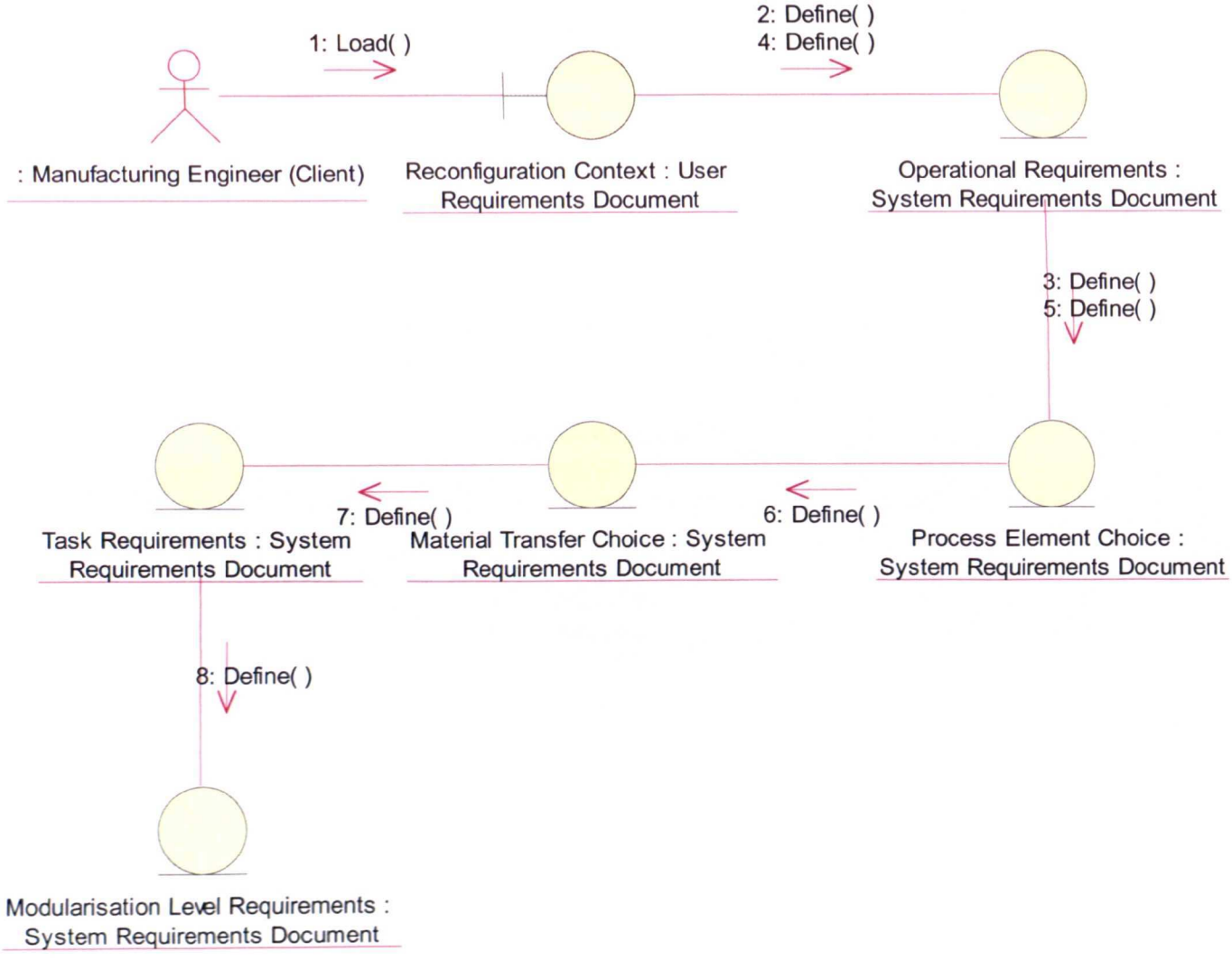
Decision Tree for System Requirements Specification



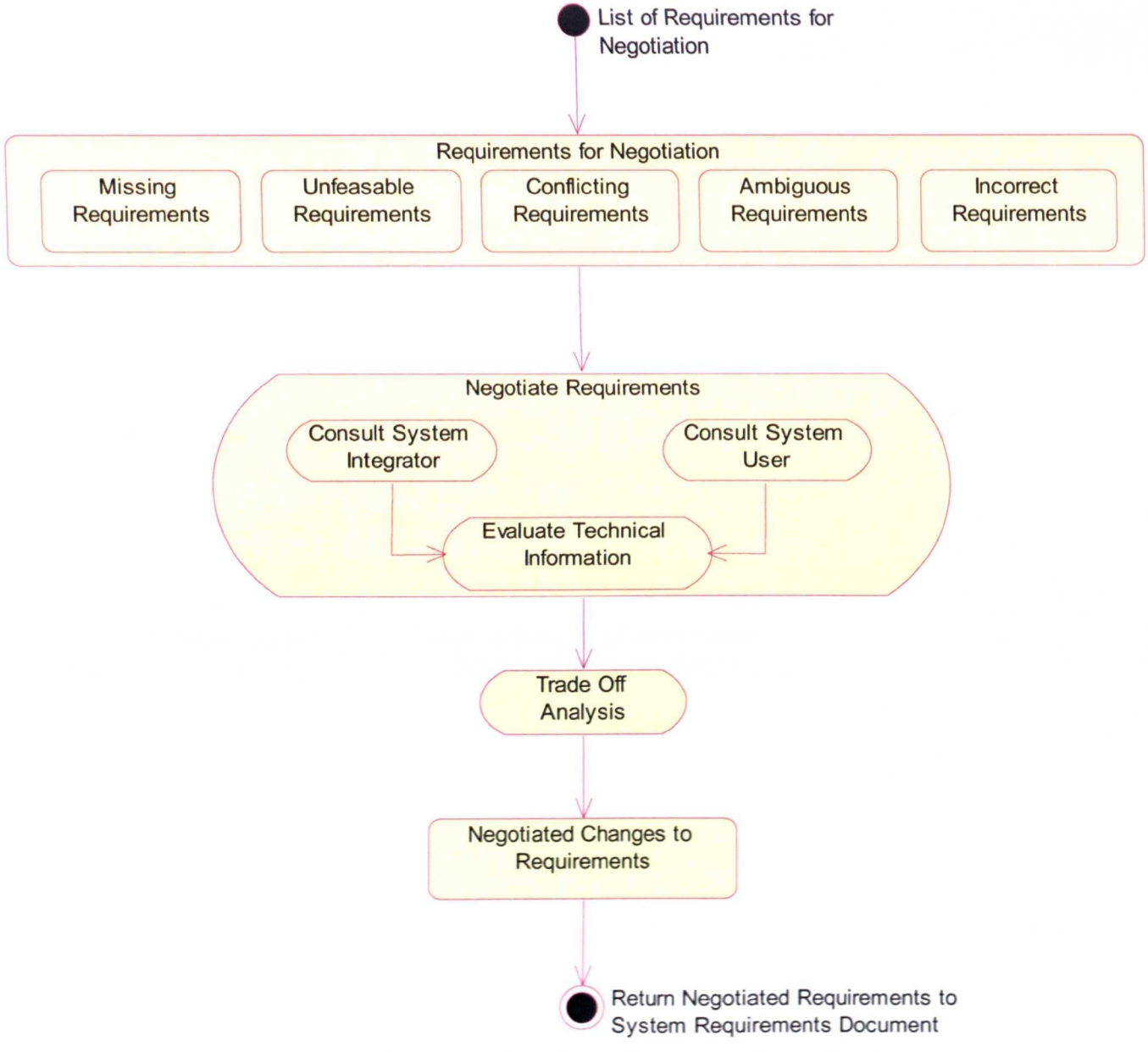
Activity Diagram for System Requirements Specification



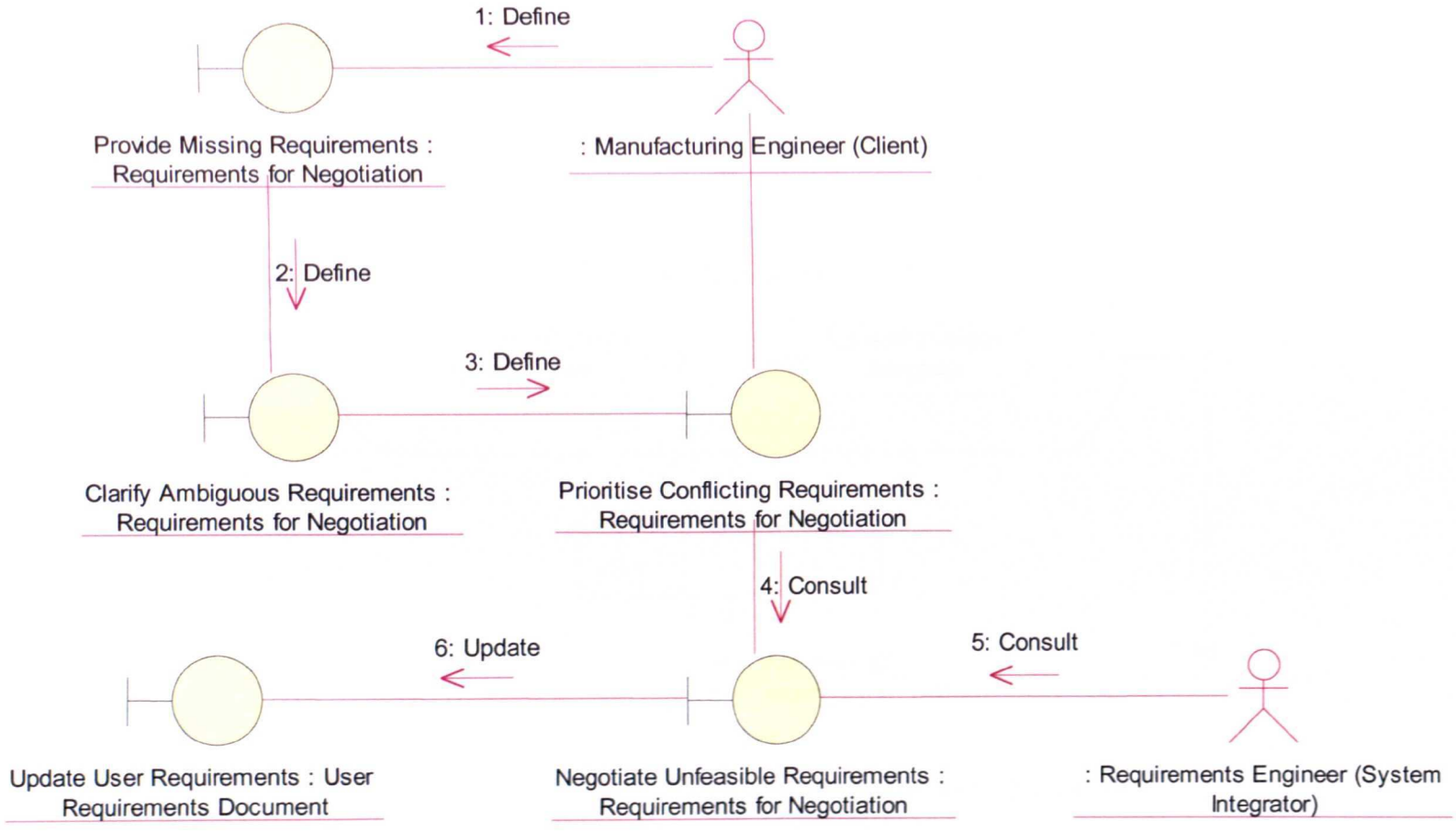
Collaboration Diagram for System Requirements Specification



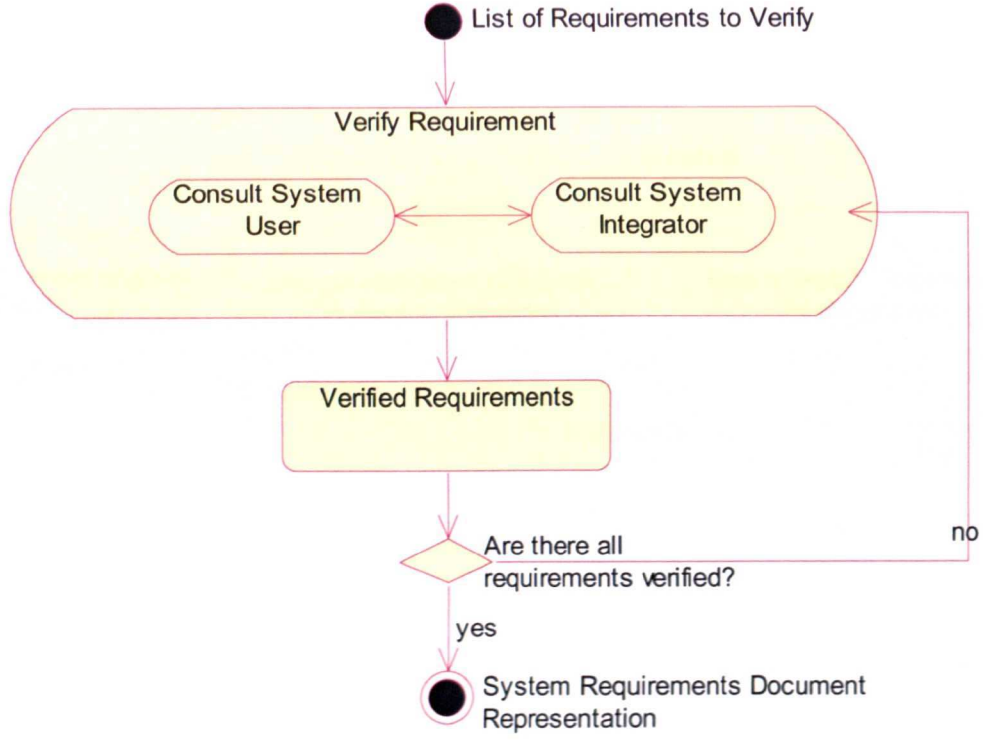
Collaboration Diagram for System Requirements Specification (System Reconfiguration)



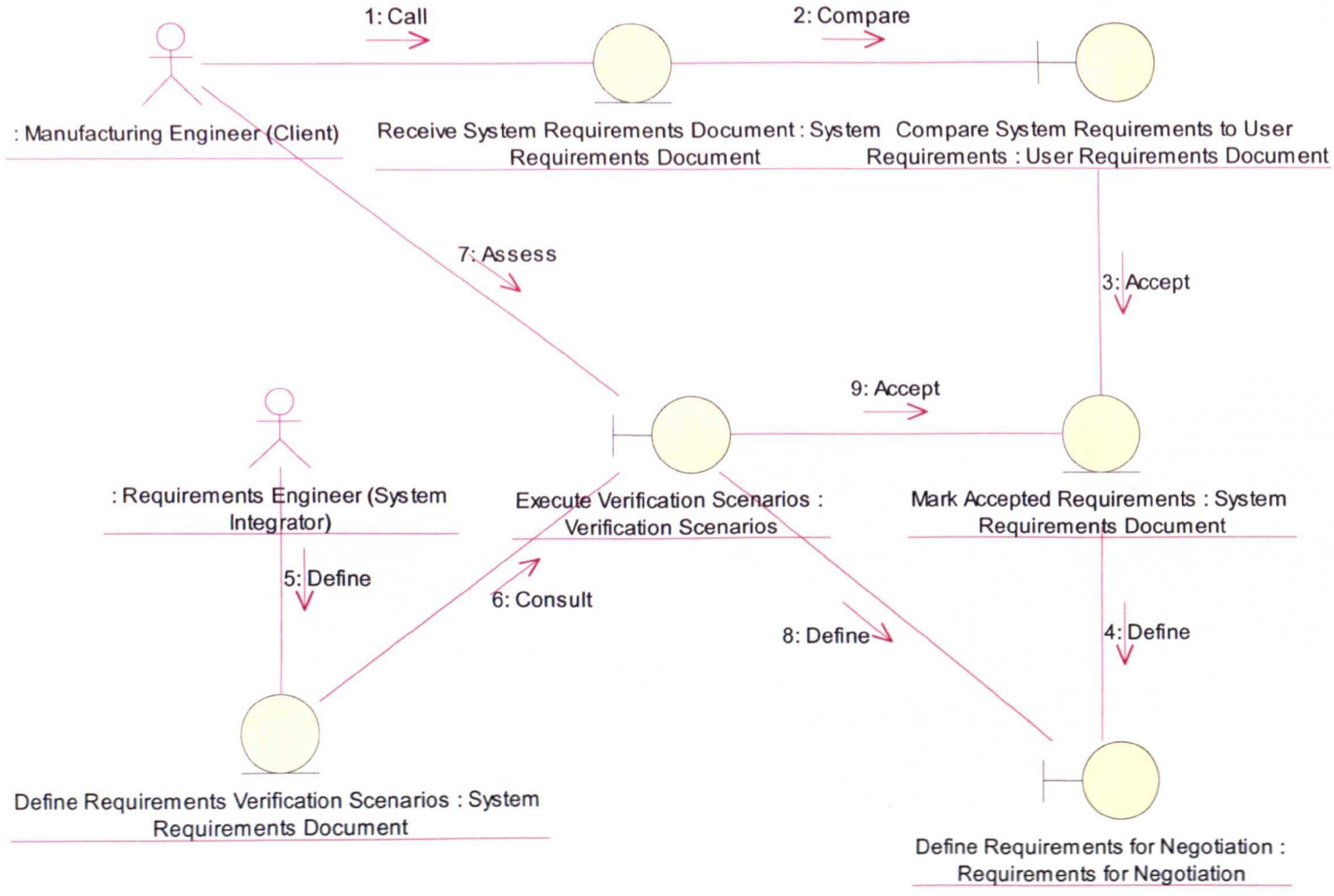
Activity Diagram for Requirements Negotiation



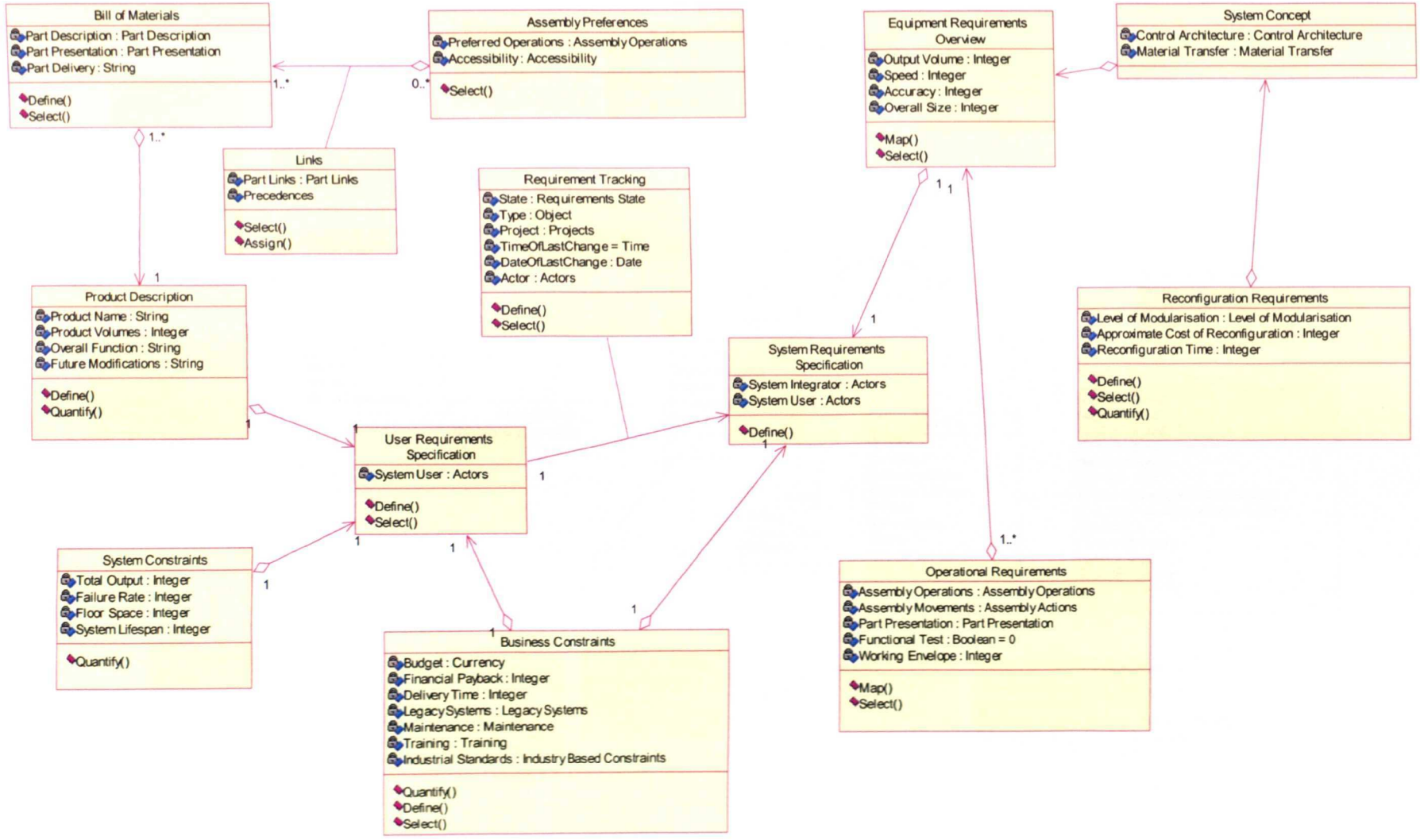
Collaboration Diagram for Requirements Negotiation



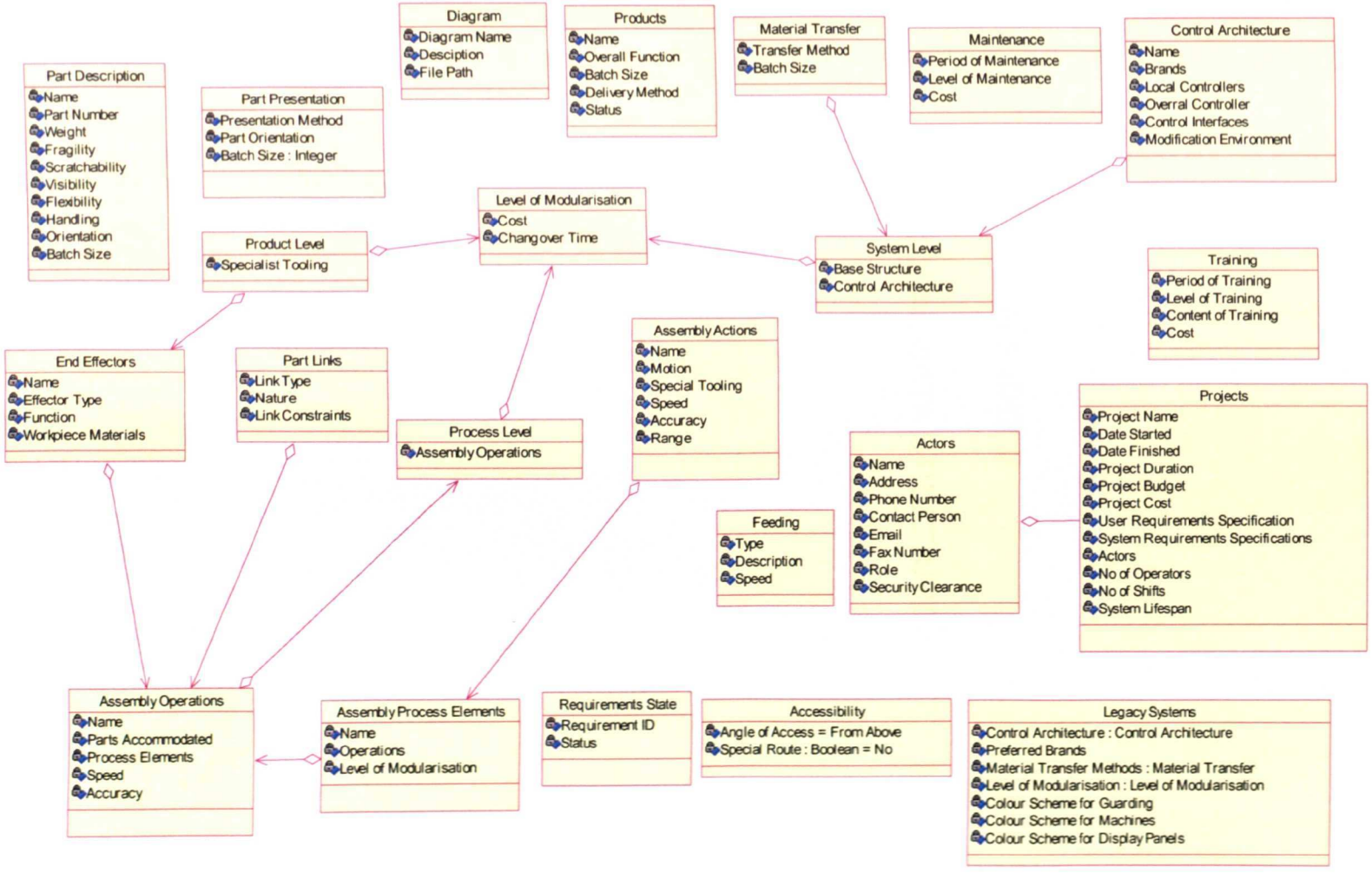
Activity Diagram for Requirements Verification



Collaboration Diagram for Requirements Verification



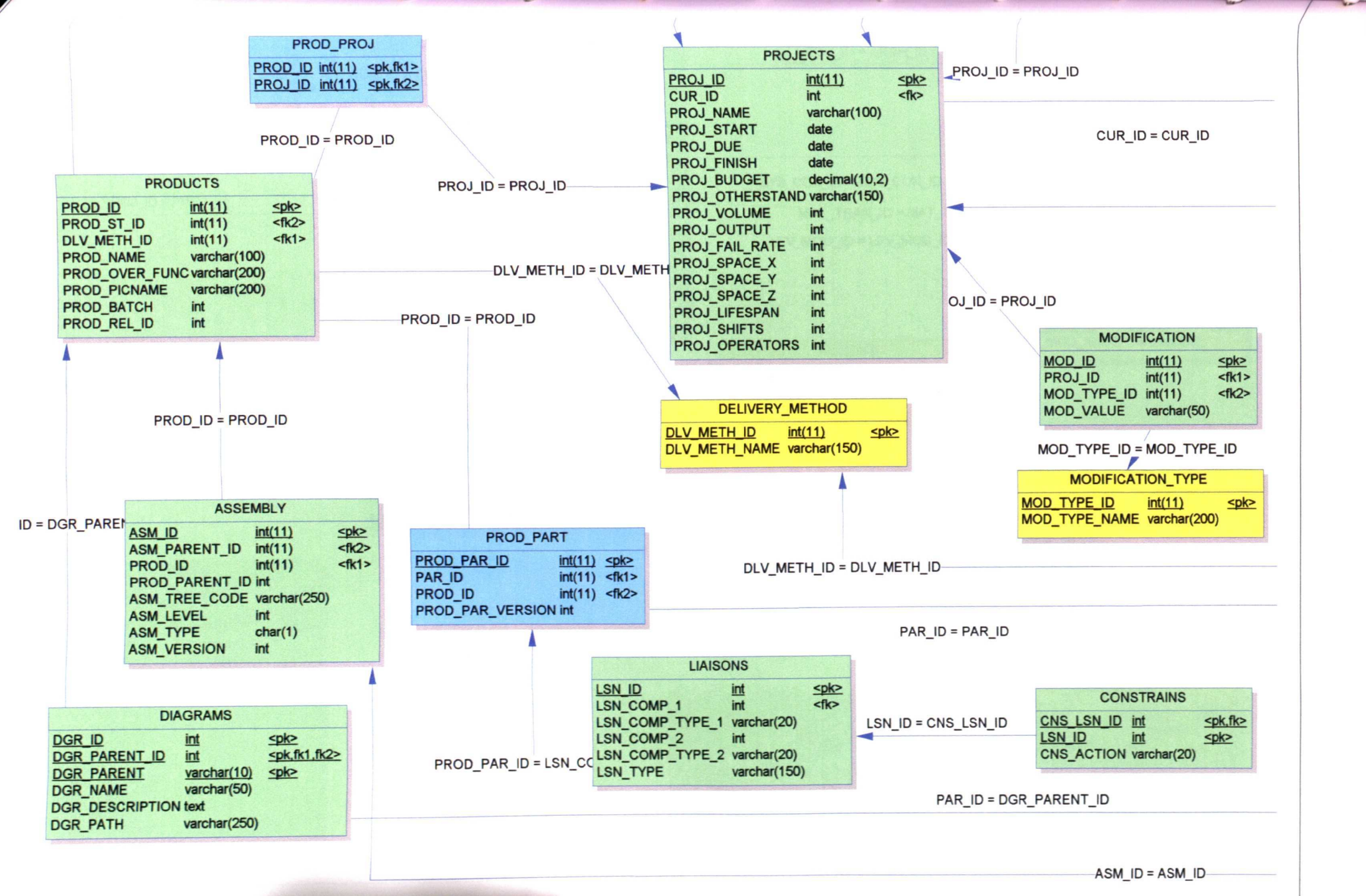
Summary of Requirements Document Classes

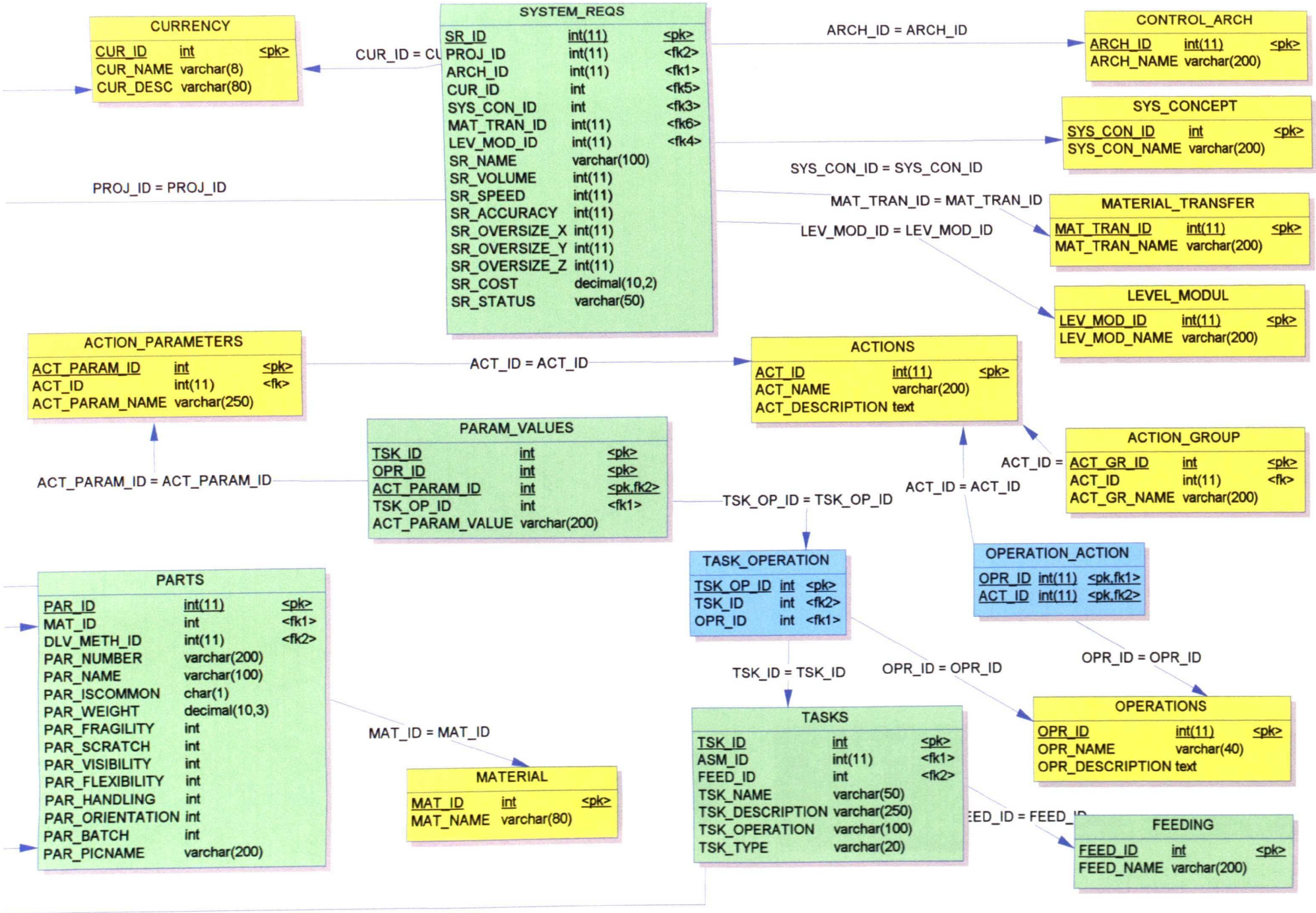


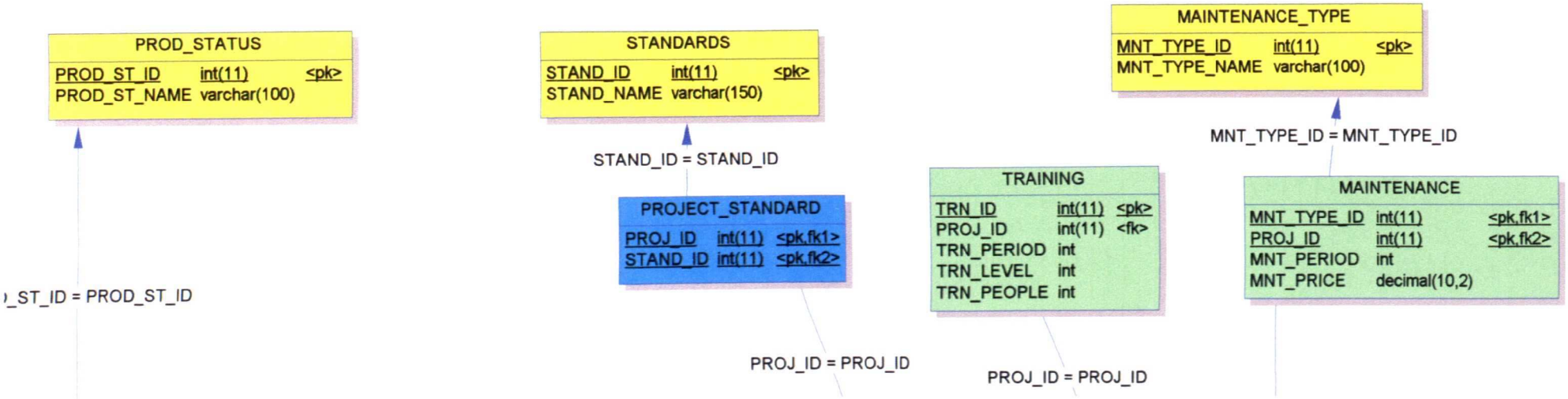
Summary of Requirements Document Sub Classes

APPENDIX B

RELATIONAL DATA STRUCTURE


PROD_PROJ
PROD_ID int(11) <pk,fk1>
PROJ_ID int(11) <pk,fk2>
PROJECTS
PROJ_ID int(11) <pk>
CUR_ID int <fk>
PROJ_NAME varchar(100)
PROJ_START date
PROJ_DUE date
PROJ_FINISH date
PROJ_BUDGET decimal(10,2)
PROJ_OTHERSTAND varchar(150)
PROJ_VOLUME int
PROJ_OUTPUT int
PROJ_FAIL_RATE int
PROJ_SPACE_X int
PROJ_SPACE_Y int
PROJ_SPACE_Z int
PROJ_LIFESPAN int
PROJ_SHIFTS int
PROJ_OPERATORS int
PROJ_ID = PROJ_ID
CUR_ID = CUR_ID
PROD_ID = PROD_ID
PROJ_ID = PROJ_ID
PRODUCTS
PROD_ID int(11) <pk>
PROD_ST_ID int(11) <fk2>
DLV_METH_ID int(11) <fk1>
PROD_NAME varchar(100)
PROD_OVER_FUNC varchar(200)
PROD_PICNAME varchar(200)
PROD_BATCH int
PROD_REL_ID int
DLV_METH_ID = DLV_METH
OJ_ID = PROJ_ID
PROD_ID = PROD_ID
MODIFICATION
MOD_ID int(11) <pk>
PROJ_ID int(11) <fk1>
MOD_TYPE_ID int(11) <fk2>
MOD_VALUE varchar(50)
DELIVERY_METHOD
DLV_METH_ID int(11) <pk>
DLV_METH_NAME varchar(150)
PROD_ID = PROD_ID
MOD_TYPE_ID = MOD_TYPE_ID
MODIFICATION_TYPE
MOD_TYPE_ID int(11) <pk>
MOD_TYPE_NAME varchar(200)
ID = DGR_PAREN
ASSEMBLY
ASM_ID int(11) <pk>
ASM_PARENT_ID int(11) <fk2>
PROD_ID int(11) <fk1>
PROD_PARENT_ID int
ASM_TREE_CODE varchar(250)
ASM_LEVEL int
ASM_TYPE char(1)
ASM_VERSION int
PROD_PART
PROD_PAR_ID int(11) <pk>
PAR_ID int(11) <fk1>
PROD_ID int(11) <fk2>
PROD_PAR_VERSION int
DLV_METH_ID = DLV_METH_ID
PAR_ID = PAR_ID
LIAISONS
LSN_ID int <pk>
LSN_COMP_1 int <fk>
LSN_COMP_TYPE_1 varchar(20)
LSN_COMP_2 int
LSN_COMP_TYPE_2 varchar(20)
LSN_TYPE varchar(150)
CONSTRAINS
CNS_LSN_ID int <pk,fk>
LSN_ID int <pk>
CNS_ACTION varchar(20)
LSN_ID = CNS_LSN_ID
DIAGRAMS
DGR_ID int <pk>
DGR_PARENT_ID int <pk,fk1,fk2>
DGR_PARENT varchar(10) <pk>
DGR_NAME varchar(50)
DGR_DESCRIPTION text
DGR_PATH varchar(250)
PROD_PAR_ID = LSN_CC
PAR_ID = DGR_PARENT_ID
ASM_ID = ASM_ID


CURRENCY
CUR_ID int <pk>
CUR_NAME varchar(8)
CUR_DESC varchar(80)
CUR_ID = CU
SYSTEM_REQS
SR_ID int(11) <pk>
PROJ_ID int(11) <fk2>
ARCH_ID int(11) <fk1>
CUR_ID int <fk5>
SYS_CON_ID int <fk3>
MAT_TRAN_ID int(11) <fk6>
LEV_MOD_ID int(11) <fk4>
SR_NAME varchar(100)
SR_VOLUME int(11)
SR_SPEED int(11)
SR_ACCURACY int(11)
SR_OVERSIZE_X int(11)
SR_OVERSIZE_Y int(11)
SR_OVERSIZE_Z int(11)
SR_COST decimal(10,2)
SR_STATUS varchar(50)
ARCH_ID = ARCH_ID
CONTROL_ARCH
ARCH_ID int(11) <pk>
ARCH_NAME varchar(200)
SYS_CONCEPT
SYS_CON_ID int <pk>
SYS_CON_NAME varchar(200)
SYS_CON_ID = SYS_CON_ID
PROJ_ID = PROJ_ID
MAT_TRAN_ID = MAT_TRAN_ID
MATERIAL_TRANSFER
MAT_TRAN_ID int(11) <pk>
MAT_TRAN_NAME varchar(200)
LEV_MOD_ID = LEV_MOD_ID
LEVEL_MODUL
LEV_MOD_ID int(11) <pk>
LEV_MOD_NAME varchar(200)
ACTION_PARAMETERS
ACT_PARAM_ID int <pk>
ACT_ID int(11) <fk>
ACT_PARAM_NAME varchar(250)
ACT_ID = ACT_ID
ACTIONS
ACT_ID int(11) <pk>
ACT_NAME varchar(200)
ACT_DESCRIPTION text
ACT_ID =
ACTION_GROUP
ACT_GR_ID int <pk>
ACT_ID int(11) <fk>
ACT_GR_NAME varchar(200)
ACT_PARAM_ID = ACT_PARAM_ID
PARAM_VALUES
TSK_ID int <pk>
OPR_ID int <pk>
ACT_PARAM_ID int <pk,fk2>
TSK_OP_ID int <fk1>
ACT_PARAM_VALUE varchar(200)
TSK_OP_ID = TSK_OP_ID
ACT_ID = ACT_ID
TASK_OPERATION
TSK_OP_ID int <pk>
TSK_ID int <fk2>
OPR_ID int <fk1>
OPERATION_ACTION
OPR_ID int(11) <pk,fk1>
ACT_ID int(11) <pk,fk2>
PARTS
PAR_ID int(11) <pk>
MAT_ID int <fk1>
DLV_METH_ID int(11) <fk2>
PAR_NUMBER varchar(200)
PAR_NAME varchar(100)
PAR_ISCOMMON char(1)
PAR_WEIGHT decimal(10,3)
PAR_FRAGILITY int
PAR_SCRATCH int
PAR_VISIBILITY int
PAR_FLEXIBILITY int
PAR_HANDLING int
PAR_ORIENTATION int
PAR_BATCH int
PAR_PICNAME varchar(200)
TSK_ID = TSK_ID
OPR_ID = OPR_ID
OPR_ID = OPR_ID
OPERATIONS
OPR_ID int(11) <pk>
OPR_NAME varchar(40)
OPR_DESCRIPTION text
MAT_ID = MAT_ID
MATERIAL
MAT_ID int <pk>
MAT_NAME varchar(80)
TASKS
TSK_ID int <pk>
ASM_ID int(11) <fk1>
FEED_ID int <fk2>
TSK_NAME varchar(50)
TSK_DESCRIPTION varchar(250)
TSK_OPERATION varchar(100)
TSK_TYPE varchar(20)
EED_ID = FEED_ID
FEEDING
FEED_ID int <pk>
FEED_NAME varchar(200)


PROD_STATUS
PROD_ST_ID int(11) <pk>
PROD_ST_NAME varchar(100)
_ST_ID = PROD_ST_ID
STANDARDS
STAND_ID int(11) <pk>
STAND_NAME varchar(150)
STAND_ID = STAND_ID
PROJECT_STANDARD
PROJ_ID int(11) <pk,fk1>
STAND_ID int(11) <pk,fk2>
PROJ_ID = PROJ_ID
TRAINING
TRN_ID int(11) <pk>
PROJ_ID int(11) <fk>
TRN_PERIOD int
TRN_LEVEL int
TRN_PEOPLE int
PROJ_ID = PROJ_ID
MAINTENANCE_TYPE
MNT_TYPE_ID int(11) <pk>
MNT_TYPE_NAME varchar(100)
MNT_TYPE_ID = MNT_TYPE_ID
MAINTENANCE
MNT_TYPE_ID int(11) <pk,fk1>
PROJ_ID int(11) <pk,fk2>
MNT_PERIOD int
MNT_PRICE decimal(10,2)

APPENDIX C

SUMMARY OF MAPPING RULES

FROM RULE BASE

User Requirements	Rules	If	Then	System Requirements
Budget	Approximate cost must not exceed the budget			Approximate Cost
Legacy Systems	Control Architecture should be compatible with existing systems			Control Architecture
	Material Transfer System should be compatible with existing systems	existing = Rotary existing = In Line Conveyor existing = Manual	Specify Rotary Specify In-Line Conveyor No limitation	Material Transfer
	System Concept should reflect companies' overall strategy	High Turnover of Products Product Parts Changing Processes Changing	Do not specify fixed assembly Product Level Modularity Process Level Modularty	System Concept
Product Name	is not processed			
Production Volume	Control Architecture should be able to cope with the volume and variety of parts being assembled	Volume > 100kph Variety <= 5 types otherwise	centralised control centralised control distributed control	Control Architecture
	Material Transfer System should be able to cope with the volume and variety of parts being assembled	Cycle Time < 4s Variety <=2 types	not rotary rotary	Material Transfer
	System Concept should accommodate the volume and variety of parts being assembled	High Turnover of Products Product Parts Changing Processes Changing	Do not specify fixed assembly Product Level Modularity Process Level Modularty	System Concept
	Output Volume = Sum of Production Volumes of all products assembled on the assembly line			Output Volume

User Requirements	Rules	If	Then	System Requirements
	Speed = movement of parts along the line when producing at the specified Production Volume			Speed
Overall Function	no processing			
Future Modifications	If there are future modifications then control architecture must be flexible	FM = yes	CA ! = Centralised	Control Architecture
	More Flexible Material Transfer if there are Future Modifications	FM = yes	MT = Modular conveyor	Material Transfer
	Not Fixed Automation if there are Future Modifications.	FM = yes	Reconfigurable Assembly System	System Concept
	LoM = None of no future modifications LoM = Product if minor changes are made to products LoM = Process if product changes result in process changes LoM = System if changes in products are significant and/or volume fluctuations are large	FM = None Minor changes to products Changes in Processes Significant product/volume changes	No Modularity Required Product Level Modularity Process Level Modularity System Level Modularity	Level of Modularisation
Total Output	Control Architecture Must support Total Output Level	Volume > 100kph	centralised control	Control Architecture
	Material Transfer must have capacity => Total Output			Material Transfer
	Output Volume = Total Output of Line + Failure Rate			Output Volume
	System Concept must include enough capacity to support Total Output			System Concept
	Speed = movement of parts along the line when producing at the specified Total Output			Speed
Acceptable Failure Rate	Output Volume = Total Output of Line + Failure Rate			Output Volume
	Accuracy Needed = Acceptable Failure Rate			Accuracy Needed

User Requirements	Rules	If	Then	System Requirements
Floor Space	Floorspace = Overall Size			Overall Size
System Lifespan	Longer System Lifespan = More distributed control unless product and production is forecasted to be constant			Control Architecture
	Longer Lifespan = more flexible/reconfigurable system concept unless product and production will be constant			System Concept
	Longer Lifespan = Higher Level of Modularisation			Level of Modularisation
Part Links	Assembly Operation must be equivalent to part liaison constraints			Assembly Operations
Training	no processing			Training
Maintenance	no processing			Maintenance
Product Delivery	Packaging must reflect the user reqs for product delivery, e.g., bulk packed boxing, tray mounted products, etc.			Packaging
	Material transfer must accommodate the product delivery method chosen (how the product will leave the line			Material Transfer
Environmental Constraints	All Equipment Must be compatible with the Electrical power supply All Pneumatic devices should use the available compressed air source All Equipment must operate within the specified temprerature All Equipment must operate within the specified Humidity All Equipment must adhere to the floor loading requirements All Equipment Must operate within the required cycle time			All Electrically Powered Equipment All Pneumatic Devices All Equipment All Equipment All Equipment All Equipment

User Requirements	Rules	If	Then	System Requirements
	All Equipment must be within the noise level baseline			All Equipment
Cycle Time Constraints	Choose relevant assembly concepts according to cycle time required	Cycle Time => 4s	conventional assembly	System Concept
		Cycle Time < 4s AND rate is consistently high for all parts	Continuous Motion System	System Concept
		Cycle Time < 4s AND rate is variable between parts	Double Up Processes	System Concept
Part Links	Choose Correct Assembly Operation according to Part Link characteristic	part link = glued	operation = adhesive bonding	Assembly Operations
		part link = insert part link = tight fit part link = screwed	operation = insertion operation = press in operation = screwing	
		part link = snap fitted	operation = snap fitting	
		part link = welded/soldered part link = rivetted	operation = welding/soldering operation = rivetting	
Parts (All Data)	Feeding - when part supply is pre-defined	part is manufactured on-line	feeding = on-line manufacturing	
		part is supplied in pallets	feeding = pallet feeding	
		part is supplied in tape	feeding = tape feeding	
		part is supplied in bulk bags/boxes	feeding = bowl feeding	
		part is liquid	feeding = Liquid Dispensing	
		part is supplied on film	feeding = film feeding	
		part is supplied in foil	feeding = foil feeding	

User Requirements	Rules	If	Then	System Requirements
		part is supplied in magazines	feeding = magazine	Part Feeding
		part is supplied in trays	feeding = tray feeding	
	Feeding - when part supply is not pre-defined	assembly can be integrated into manufacturing of the part	feeding = Pallet feeding OR on-line manufacturing	
		part is manufactured on-line	feeding = on-line manufacturing	
		part is small	feeding = tape feeding	
		part is easy to scratch	feeding =! bowl feeding	
		part is easy to orientate automatically	feeding = bowl feeding	
		part is fragile	feeding = tray feeding	
		part is easy to scratch AND scratchability cannot be limited	feeding = tray feeding OR magazine feeding	
Parts (All Data)	Material Transfer Rules	parts are heavy AND/OR bulky	MTS = AGV	Material Transfer
		parts cannot be transferred automatically	MTS = Manual Handling	
		no of operators >1	MTS =! Rotary Table	
		manual stations to be introduced	MTS = In-line conveyor	
		parts = fragile	MTS = Transport within robot	
		many operations in small space	MTS = Rotary Table	

JESS Code for Requirements Mapping Between User Requirements and System Requirements

1. User Requirements is Acceptable Failure Rate

```
Jess>(defrule Acceptable-failure-rate
(acceptable-failure-rate-is ?a)
=>
(accuracy-needed-is <?a))
```

2. User Requirements is Environmental Constraints

```
Jess>(defrule Environmental-constraints-supply
(electrical-power-supply-is ?b)
=>
(equipment-electrical-power-is >= ?b))

Jess>(defrule Environmental-constraints-temperature
(temperature-is ?c)
=>
(equipment-temperature-is <= ?c))

Jess>(defrule Environmental-constraints-humidity
(humidity-is ?d)
=>
(equipment-humidity-is <= ?d))

Jess>(defrule Environmental-constraints-loading
(floor-loading-requirements-is ?e)
=>
(equipment-loading-is <=?e))

Jess>(defrule Environmental-constraints-time
(required-cycle-time-is ?f)
=>
(equipment-cycle-time-is <= ?f))

Jess>(defrule Environmental-constraints-baseline
(noise-level-baseline-is ?g)
=>
(equipment-baseline-is <= ?g))

Jess>(defrule Environmental-constraints-air-source
(available-compressed-air-source-is ?h)
=>
(pneumatic-device-air-is <= ?h))
```

3. User Requirements is Budget

```
Jess>(defrule Budget
(budget-is ?i)
=>
(cost-is <= ?i))
```

4. User Requirements is Part Link

```
Jess>(defrule Part-links-glued
(part-link-is glued)
=>
(assembly-operation-is adhesive bonding))

Jess>(defrule Part-links-insert
(part-link-is insert)
=>
(assembly-operation-is insertion))

Jess>(defrule Part-links-tight
(part-link-is tight-fit)
=>
(assembly-operation-is press-in))

Jess>(defrule Part-links-screwed
(part-link-is screwed)
=>
(assembly-operation-is screwing))

Jess>(defrule Part-links-snap
(part-link-is snap-fitted)
=>
(assembly-operation-is snap-fitting))

Jess>(defrule Part-links-welded
(part-link-is welded)
=>
(assembly-operation-is welding))

Jess>(defrule Part-links-soldered
(part-link-is soldered)
=>
(assembly-operation-is soldering))

Jess>(defrule Part-links-rivetted
(part-link-is rivetted)
=>
(assembly-operation-is rivetting))
```

5. System Requirements is Control Architecture

```
Jess>(defrule Control-architecture-volume
(?volume&: (> ?volume 100) | ?variety&: (<= ?variety 5))
=>
(control-architecture-is centralised-control))

Jess>(defrule Control-architecture-volume-otherwise
(not (or (?volume&: (> ?volume 100)) (?variety&: (<= ?variety 5)))
=>
(control-architecture-is distributed-control))

Jess>(defrule Control-architecture-modifications
(FM-is yes)
=>
(CA-is !centralised))
```

```
Jess>(defrule Control-architecture-output
(?volume&: (> ?volume 100))
=>
(control-architecture-is centralised-control))

Jess>(defrule Control-architecture-lifespan
(system-lifespan >5)
=>
(control-architecture -is distributed-control))
```

6. System Requirements is Level of Modularization

```
Jess>(defrule LoM-modifications-none
(FM-is none)
=>
(LoM-is none))

Jess>(defrule LoM-modifications-product
(product-change-is minor)
=>
(LoM-is product))

Jess>(defrule LoM-modifications-process
(process-change-is minor)
=>
(LoM-is process))

Jess>(defrule LoM-modifications-system
(product-change-is significant | volume-fluctuations-are large)
=>
(LoM-is system))

Jess>(defrule LoM-lifespan5
(lifespan-is >5)
=>
(LoM-is !product)
(LOM-is !none))

Jess>(defrule LoM-lifespan10
(lifespan-is >10)
=>
(LoM-is !process))
```

7. System Requirements is Material Transfer

```
Jess>(defrule MT-legacy-rotary
(existing-is rotary)
=>
(MT-is rotary))

Jess>(defrule MT-legacy-conveyor
(existing-is in-line-conveyor)
=>
(MT-is in-line-conveyor))

Jess>(defrule MT-legacy-manual
(existing-is manual)
```

```
=>
(MT-is no-limitation))

Jess>(defrule MT-volume
(?cycle-time&: (>= ?cycle-time 4) | ?variety&: (<= ?variety 2))
=>
(MT-is rotary))

Jess>(defrule MT-modifications
(FM-is yes)
=>
(MT-is modular-conveyor))

Jess>(defrule MT-output
(total-output-is ?i)
=>
(?MT-capacity&: (>= ?MT-capacity <=?i)))

Jess>(defrule MT-rules-weight
(parts-are heavy | parts-are bulky)
=>
(MTS-is AGV))

Jess>(defrule MT-rules-transfer
(not (parts-transfer-is automatic))
=>
(MTS-is manual-handling))

Jess>(defrule MT-rules-operator
(?operator-number&: (> ?operator-number 1))
=>
(MTS-is not-rotary-table))

Jess>(defrule MT-rules-station
(manual-stations-are introduced)
=>
(MTS-is in-line-conveyor))

Jess>(defrule MT-rules-fragile
(parts-are fragile)
=>
(MTS-is transport-within-robot))

Jess>(defrule MT-rules-space
(space-is small)
=>
(MTS-is AGV))

Jess>(defrule MT-rules-transfer
(! (parts-transfer-is automatic))
=>
(MTS-is manual-handling))

Jess>(defrule MT-rules-operator
(?operator-number&: (> ?operator-number 1))
=>
```

(MTS-is !rotary-table))

Jess>(defrule MT-rules-station
(manual-stations-are introduced)
=>
(MTS-is in-line-conveyor))

Jess>(defrule MT-rules-fragile
(parts-are fragile)
=>
(MTS-is transport-within-robot))

Jess>(defrule MT-rules-space
(space-is small)
=>
(MTS-is rotary-table))

8. System Requirements is Overall Size

Jess>(defrule overall-size
(floor-space-is ?j)
=>
(overall-size-is <=?j))

9. System Requirements is Total Output

Jess>(defrule product-delivery
(?dummy-fact&: (= ?total-output + failure-rate)
=>
(?output-volume&: (= ?dummy-fact))

10. System Requirements is Part Feeding

Jess>(defrule feeding-pre-defined-online
(part-manufactured-is on-line)
=>
(feeding-is on-line-manufacturing))

Jess>(defrule feeding-pre-defined-pallets
(part-supply-is in-pallets)
=>
(feeding-is pallet-feeding))

Jess>(defrule feeding-pre-defined-tape
(part-supply-is in-tape)
=>
(feeding-is tape-feeding))

Jess>(defrule feeding-pre-defined-bag
(part-supply-is in-bulk-bags)
=>
(feeding-is bowl-feeding))

Jess>(defrule feeding-pre-defined-liquid
(part-is liquid)
=>
(feeding-is liquid-dispensing))

```
Jess>(defrule feeding-pre-defined-film
(part-supply-is on-film)
=>
(feeding-is film-feeding))

Jess>(defrule feeding-pre-defined-foil
(part-supply-is in-foil)
=>
(feeding-is foil-feeding))

Jess>(defrule feeding-pre-defined-magazine
(part-supply-is in-magazines)
=>
(feeding-is magazine))

Jess>(defrule feeding-pre-defined-tray
(part-supply-is in-trays)
=>
(feeding-is tray-feeding))

Jess>(defrule feeding-not-defined-assembly
(assembly-is manufacturing-of-the-part)
=>
(feeding-is pallet-feeding | feeding-is on-line-manufacturing))

Jess>(defrule feeding-not-defined-online
(part-manufactured-is on-line)
=>
(feeding-is on-line-manufacturing))

Jess>(defrule feeding-not-defined-size
(part-is small)
=>
(feeding-is tape-feeding))

Jess>(defrule feeding-not-defined-scratch
          (part-is easy-to-scratch)
        =>
          (feeding-is !bowl-feeding)
        )

Jess>(defrule feeding-not-defined-orientate
(part-is easy-to-orientate-automatically)
=>
(feeding-is bowl-feeding))

Jess>(defrule feeding-not-defined-fragile
(part-is fragile)
=>
(feeding-is tray-feeding))

Jess>(defrule feeding-not-defined-scratchability
(and (part-is easy-to-scratch ) (scratchability-is unlimited))
=>
(feeding-is tray-feeding | feeding-is magazine-feeding))
```

11. System Requirements is Speed

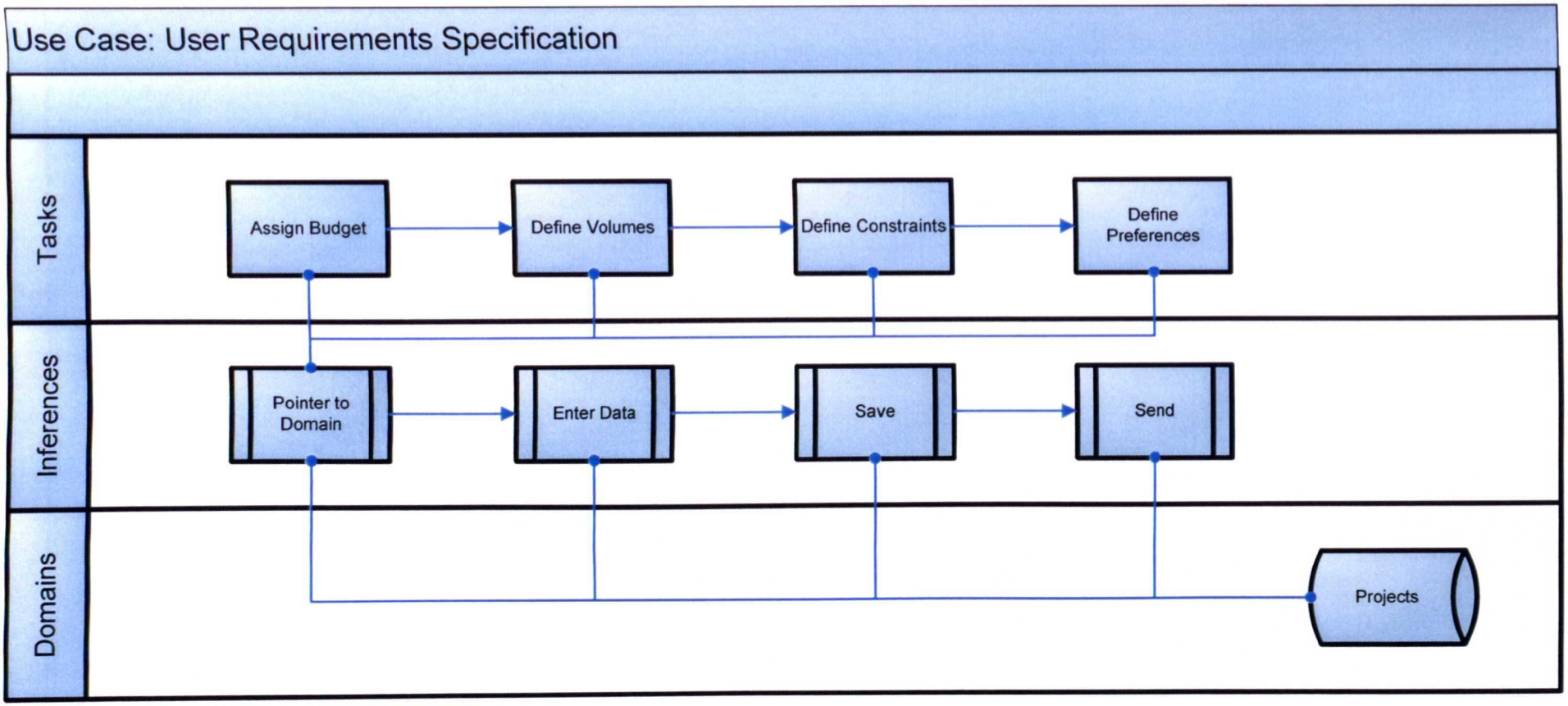
```
Jess>(defrule speed-volume
(volume-is ?k)
=>
(speed-is volume/time))
```

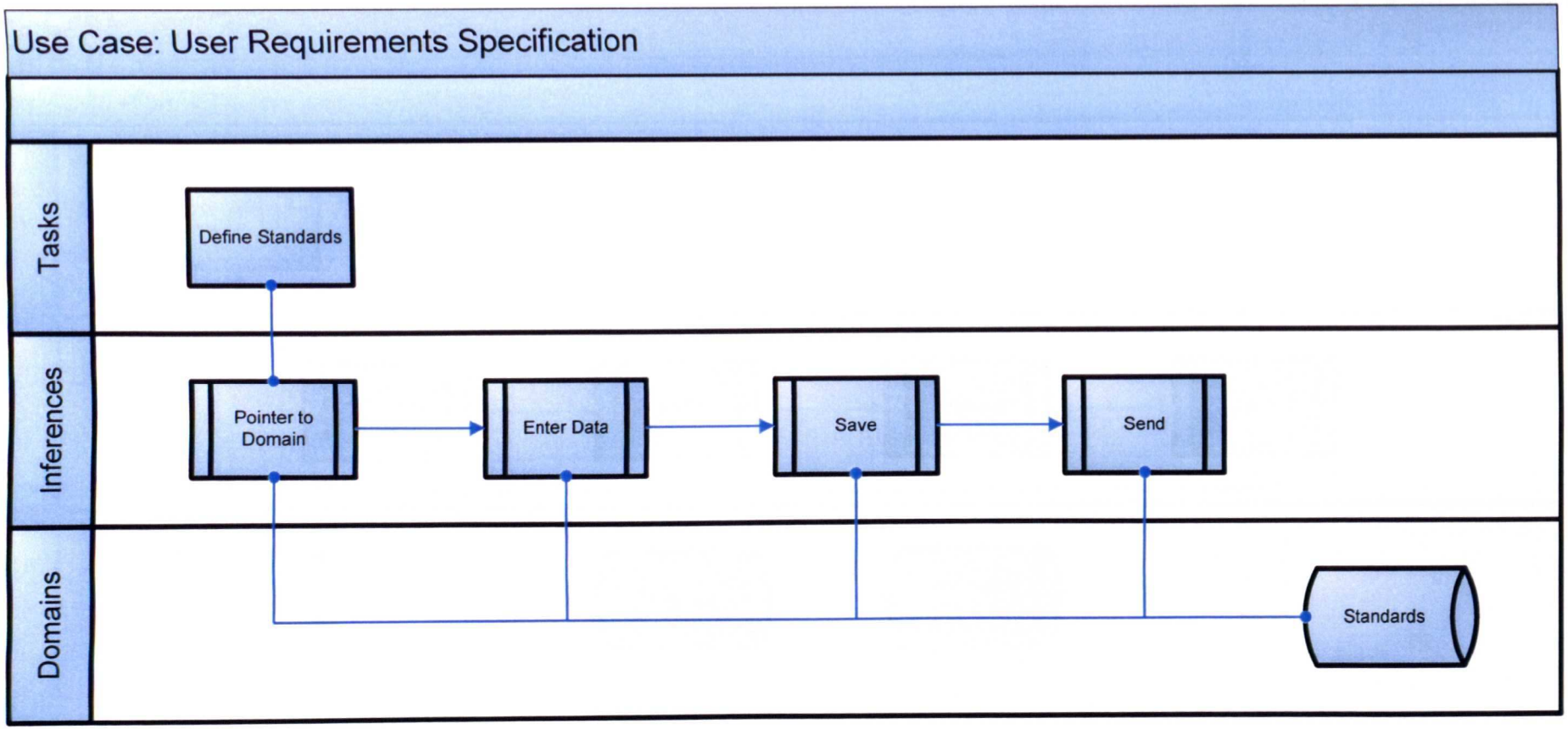
12. System Requirements is System Concept

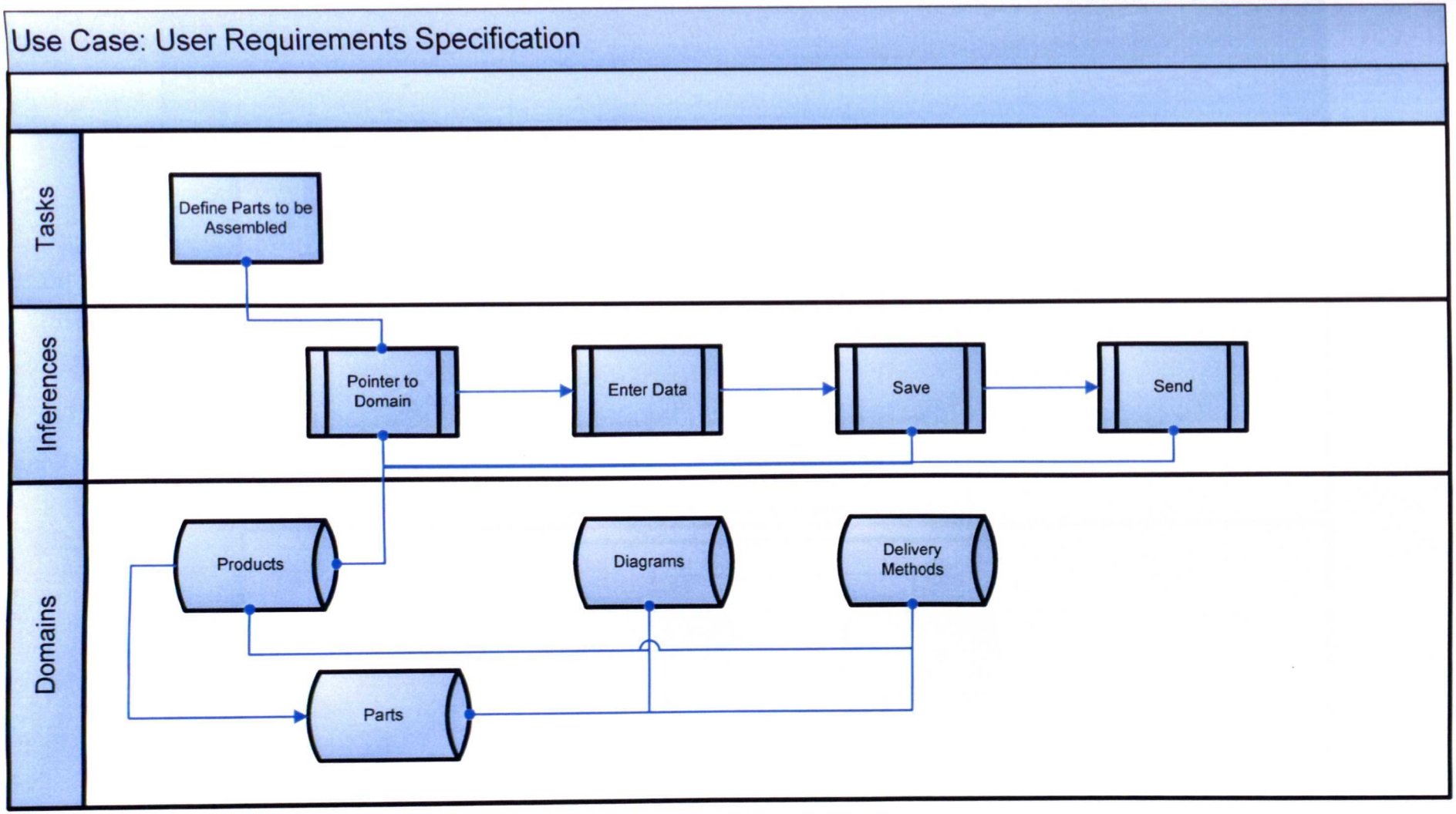
```
Jess>(defrule System-concept-legacy-turnover
(product-turnover-is high)
=>
(modularity-is !none))

Jess>(defrule System-concept-legacy-product
(product-parts-is changing)
=>
(modularity-is product-level))

Jess>(defrule System-concept-legacy-process
(process-is changing)
=>
(modularity-is process-level))

Jess>(defrule System-concept-volume-turnover
(product-turnover-is high)
=>
(modularity-is !none))

Jess>(defrule System-concept-volume-product
(product-parts-is changing)
=>
(modularity-is product-level))

Jess>(defrule System-concept-volume-process
(process-is changing)
=>
(modularity-is process-level))

Jess>(defrule System-concept-FM
(FM-is yes)
=>
(system-is reconfigurable-assembly-system))

Jess>(defrule System-concept-time-constraints-1
(?cycle-time&: (>= ?cycle-time 4))
=>
(system-is conventional-assembly))

Jess>(defrule System-concept-time-constraints-2
(and (?cycle-time&: (< ?cycle-time 4)) (rate-is high))
=>
(system-is continuous-motion-system))

Jess>(defrule System-concept-time-constraints-3
(and (?cycle-time&: (< ?cycle-time 4)) (rate-is variable))
=>
(system-is double-up-processes))
```

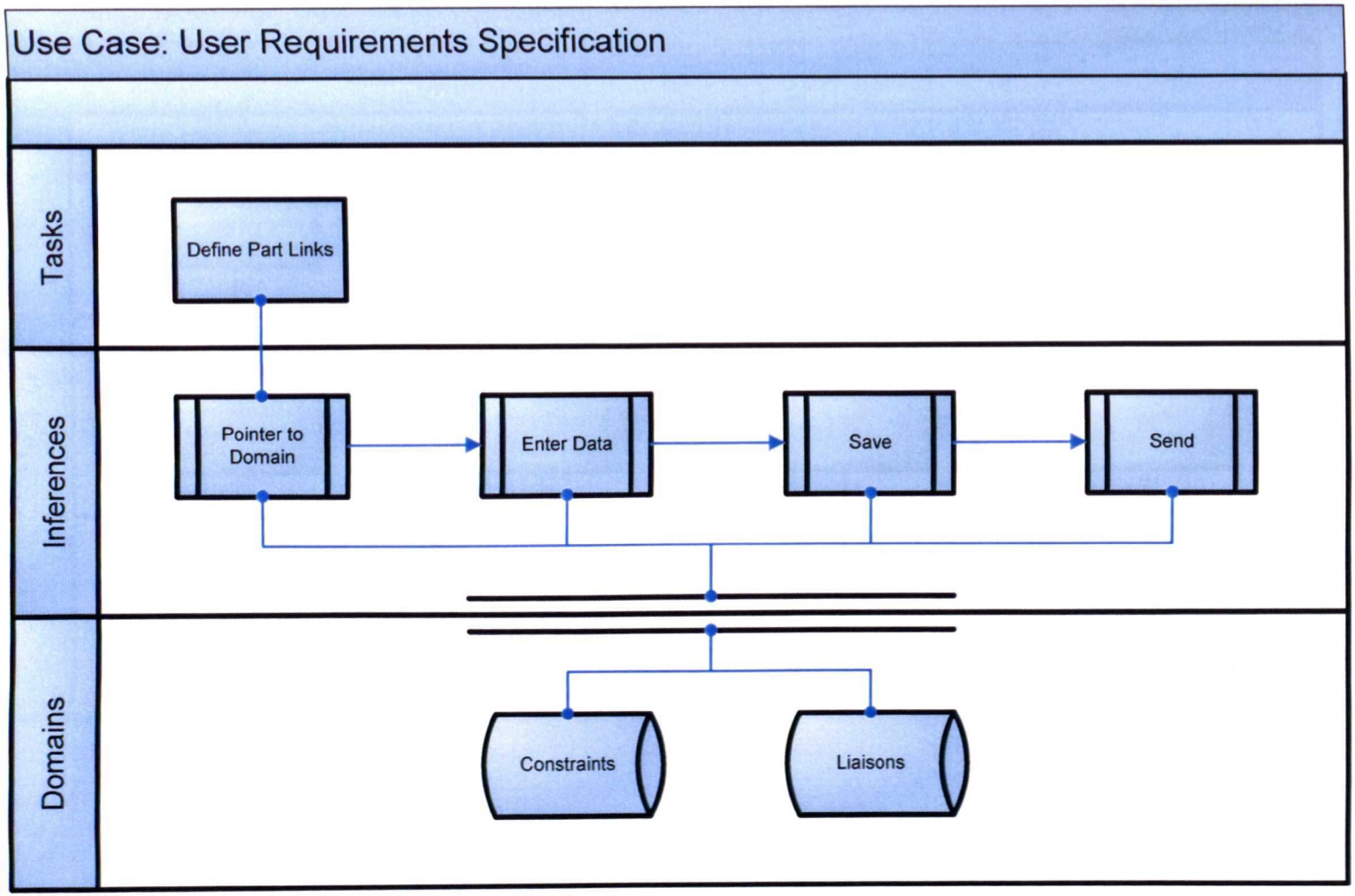
APPENDIX D

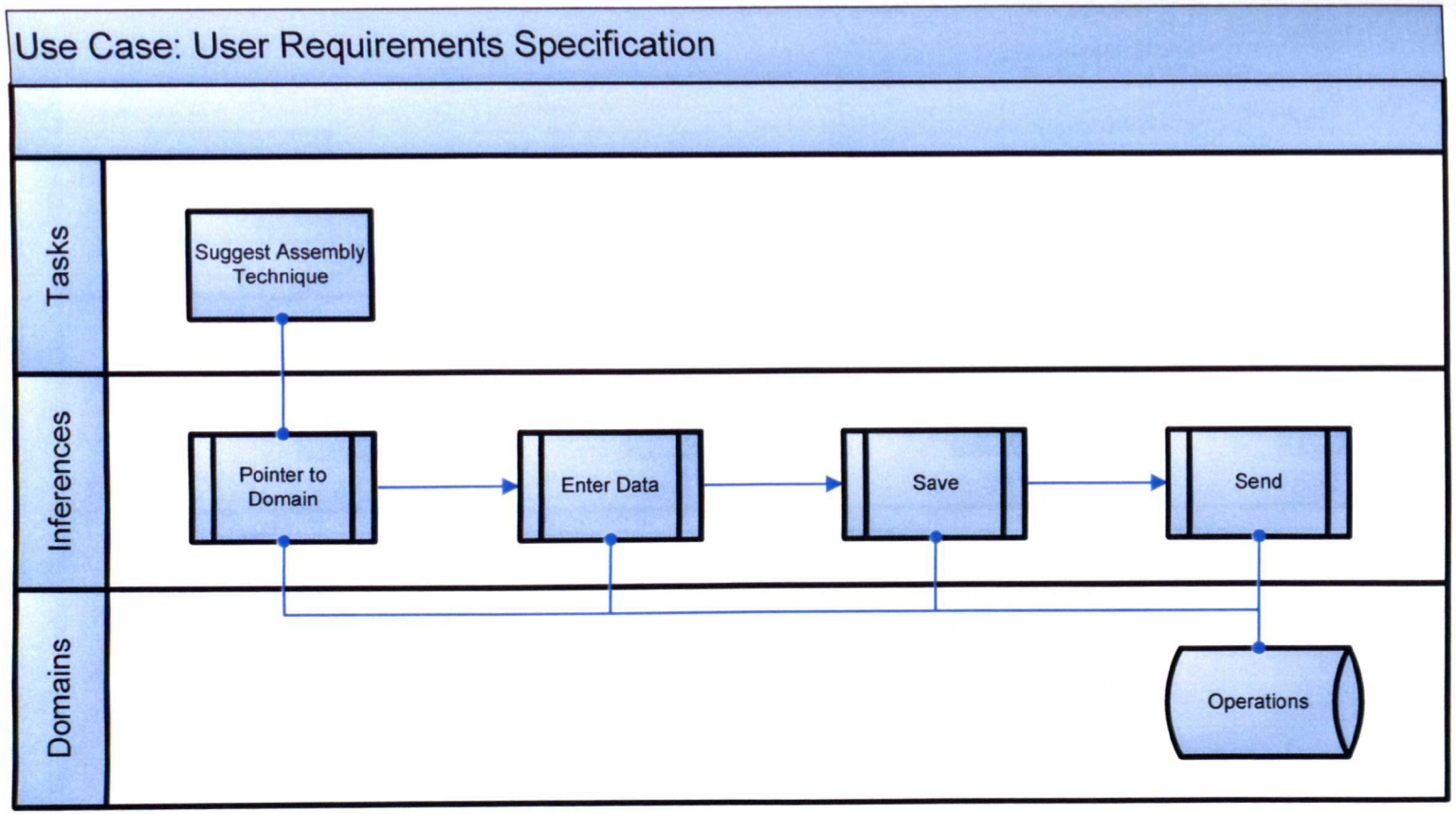
COMMONKADS DIAGRAMS WITH TASK, INFERENCE AND DOMAIN KNOWLEDGE FOR EACH USE CASE

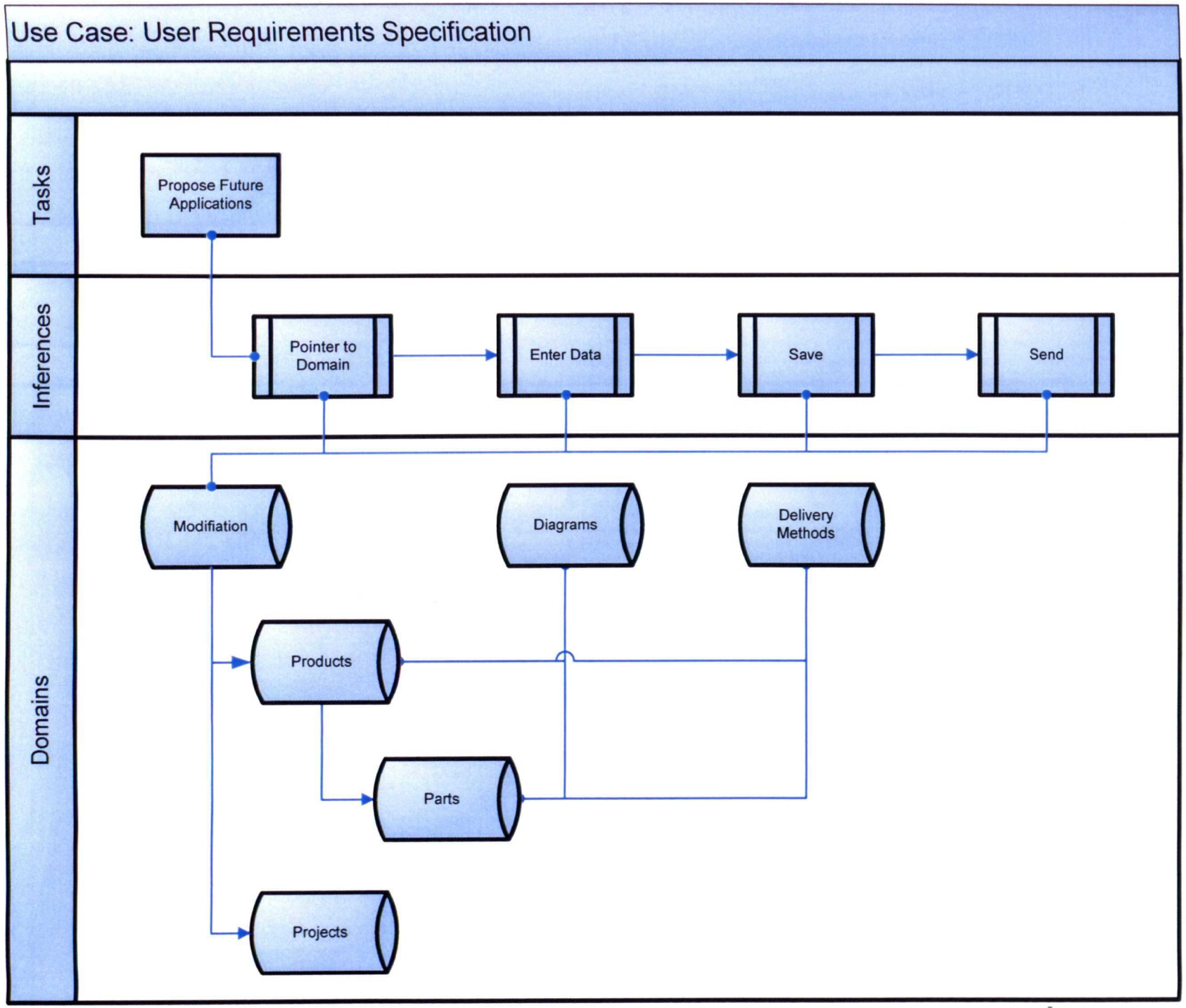

Use Case: User Requirements Specification
Tasks
Inferences
Domains
Assign Budget
Define Volumes
Define Constraints
Define Preferences
Pointer to Domain
Enter Data
Save
Send
Projects

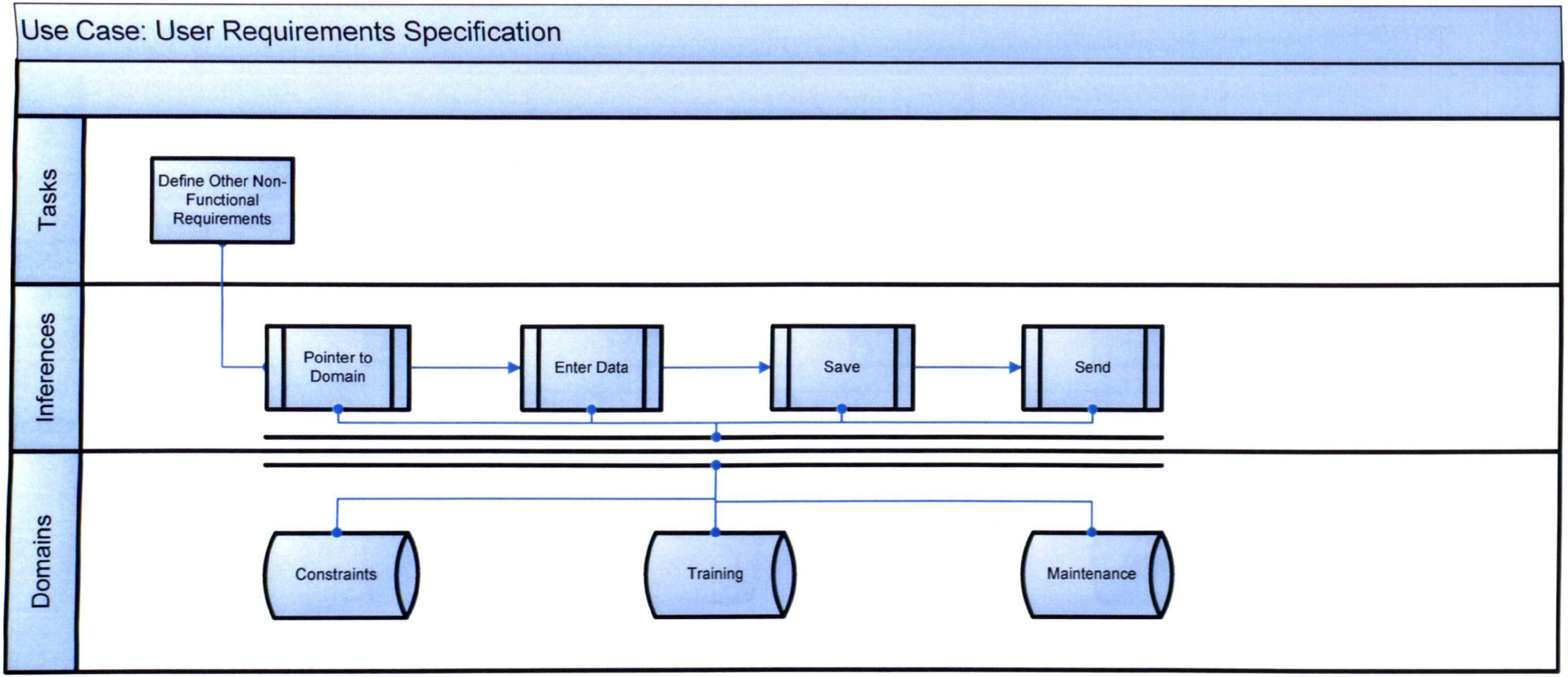

Use Case: User Requirements Specification
Tasks
Inferences
Domains
Define Standards
Pointer to Domain
Enter Data
Save
Send
Standards

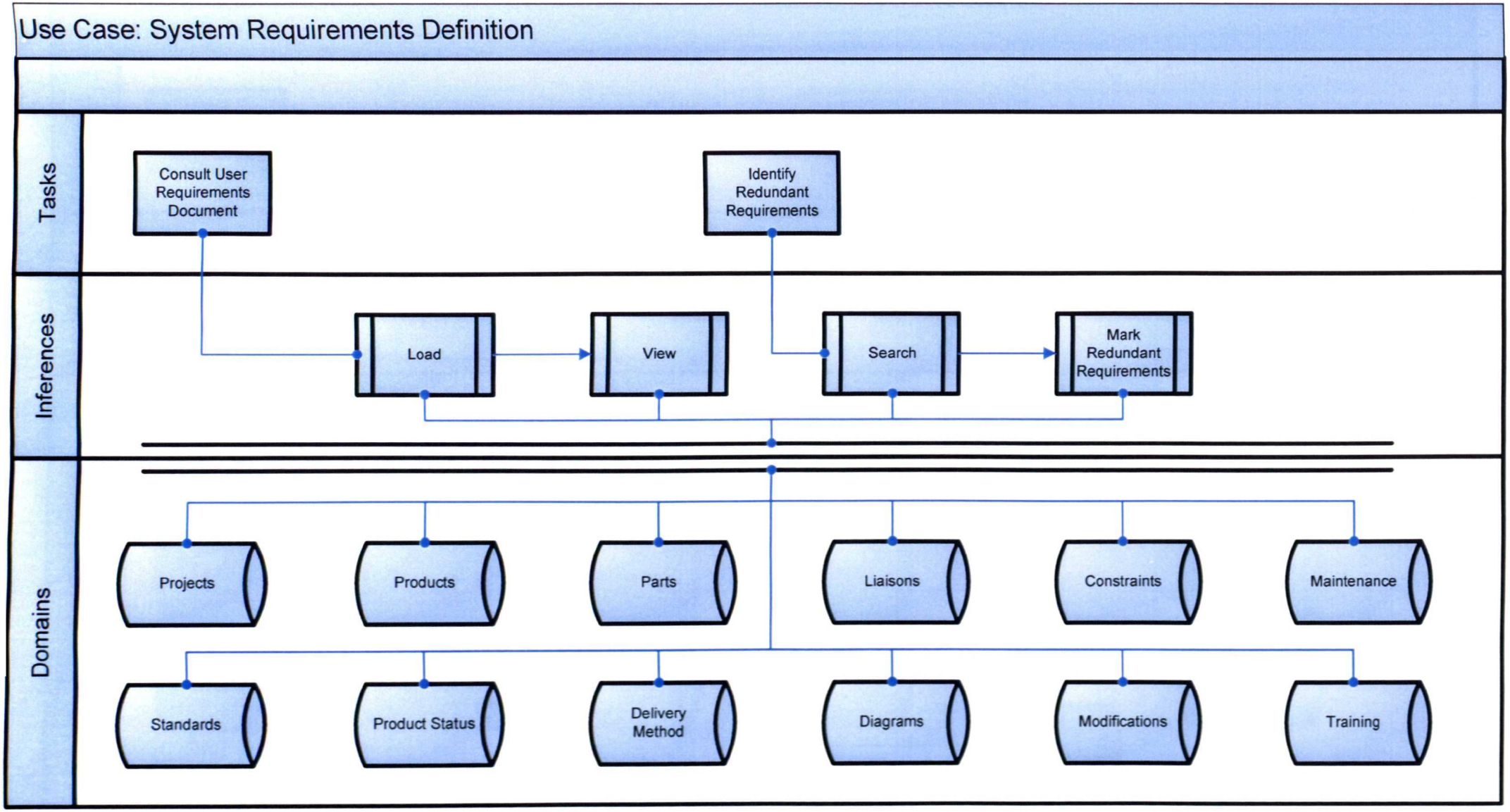

Use Case: User Requirements Specification
Tasks
Inferences
Domains
Define Parts to be Assembled
Pointer to Domain
Enter Data
Save
Send
Products
Diagrams
Delivery Methods
Parts

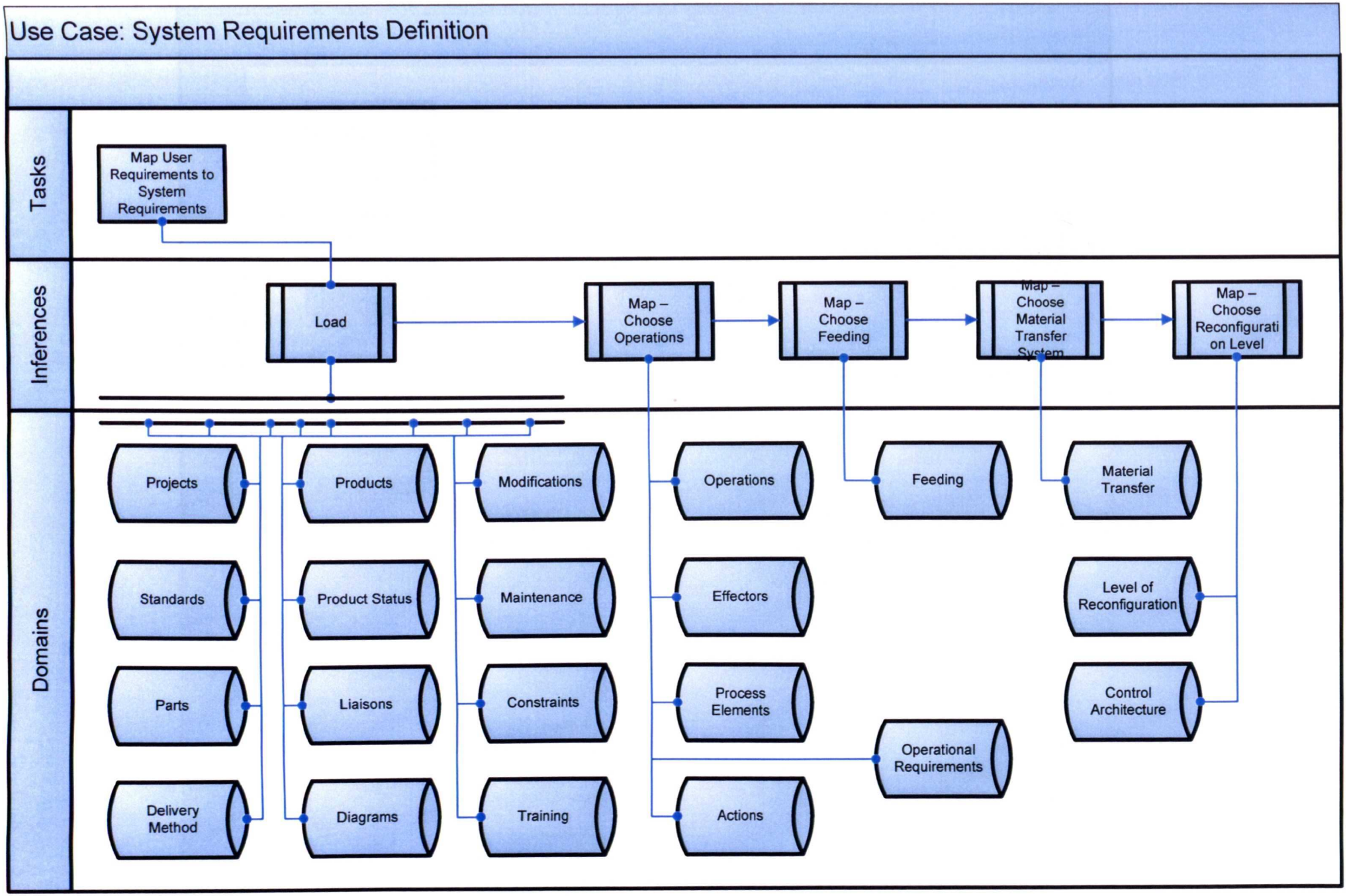

Use Case: User Requirements Specification
Tasks
Define Part Links
Inferences
Pointer to Domain
Enter Data
Save
Send
Domains
Constraints
Liaisons

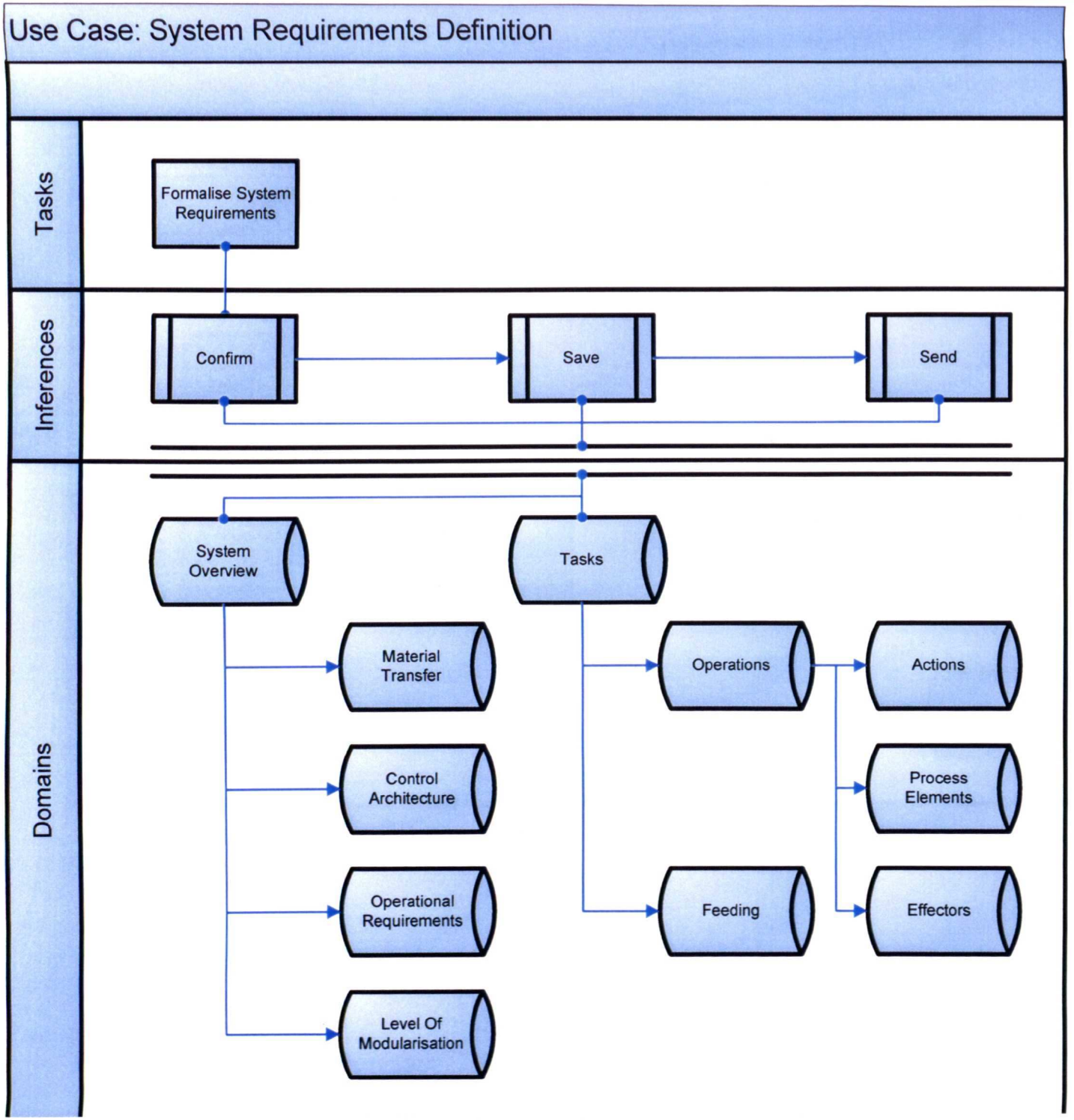

Use Case: User Requirements Specification
Tasks
Inferences
Domains
Suggest Assembly Technique
Pointer to Domain
Enter Data
Save
Send
Operations

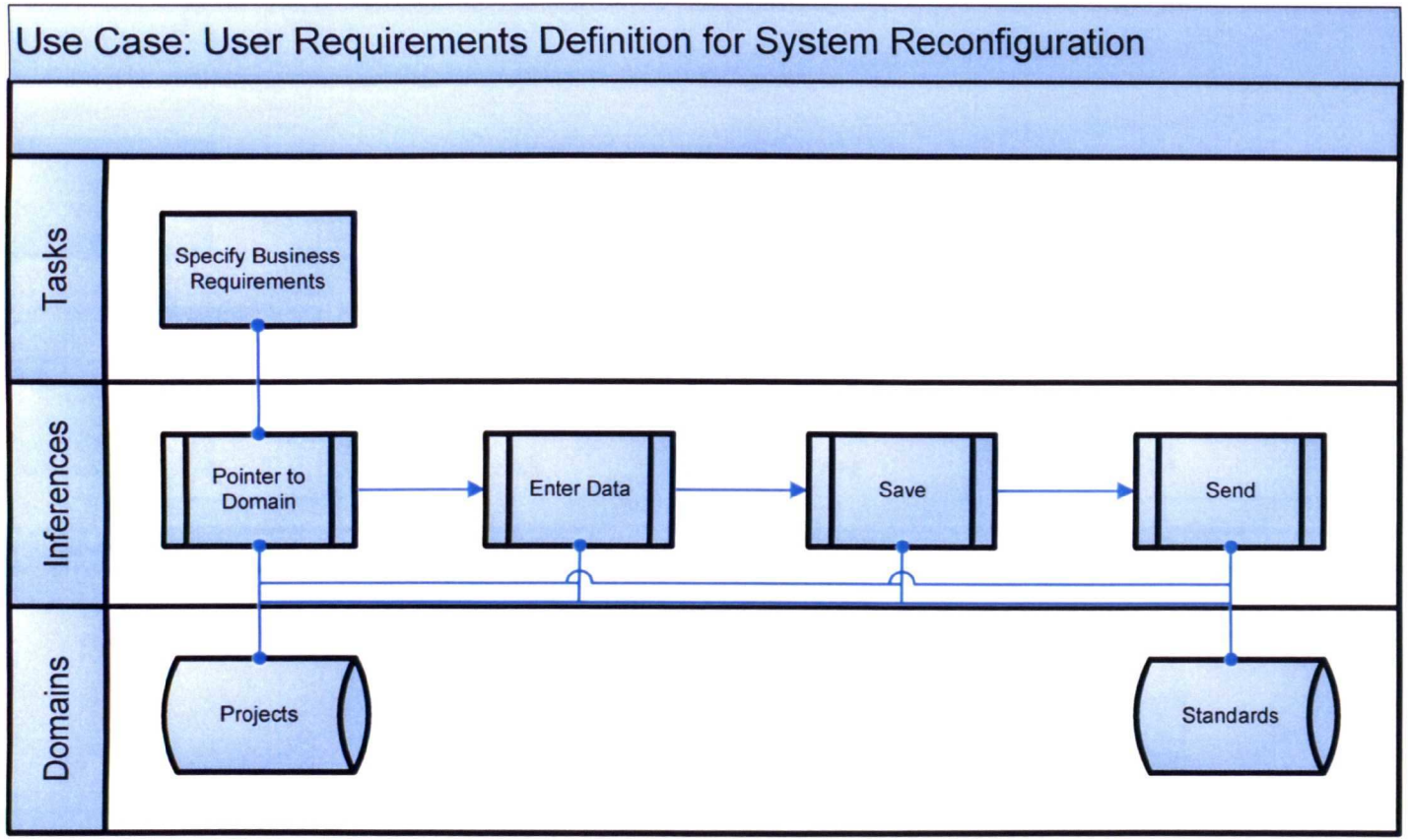

Use Case: User Requirements Specification
Tasks
Inferences
Domains
Propose Future Applications
Pointer to Domain
Enter Data
Save
Send
Modifiation
Diagrams
Delivery Methods
Products
Parts
Projects

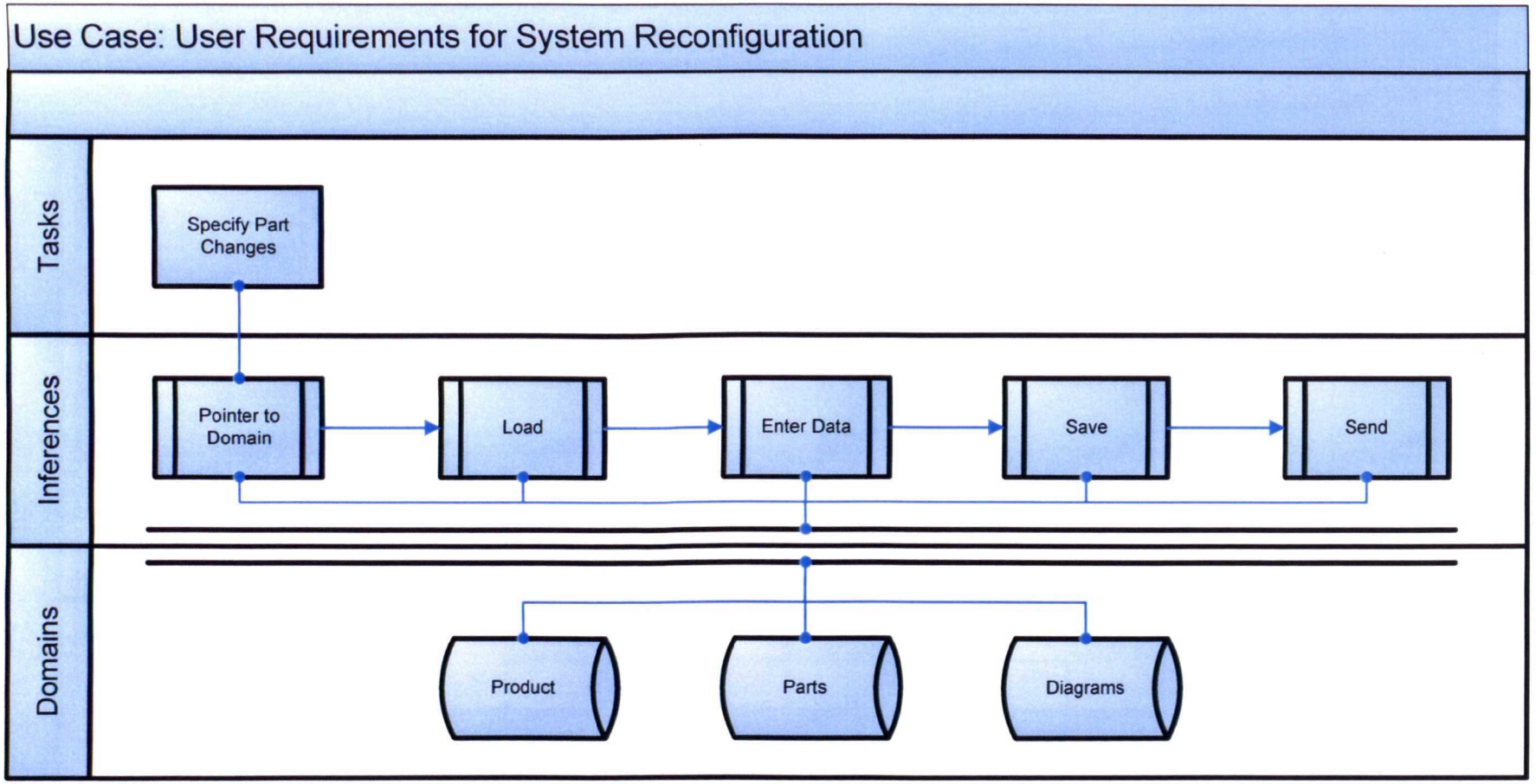

Use Case: User Requirements Specification
Tasks
Inferences
Domains
Define Other Non-Functional Requirements
Pointer to Domain
Enter Data
Save
Send
Constraints
Training
Maintenance

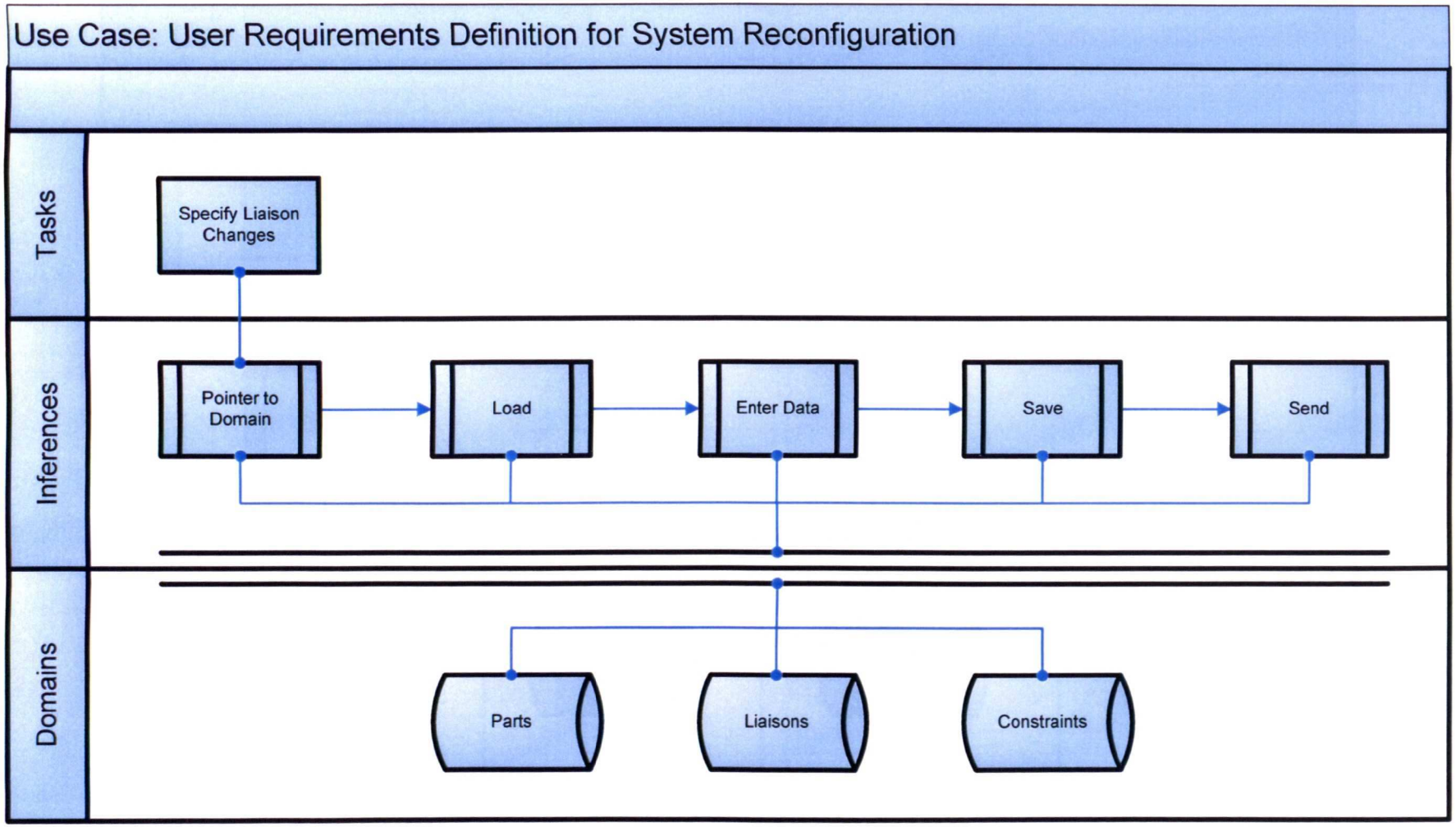

Use Case: System Requirements Definition
Tasks
Consult User Requirements Document
Identify Redundant Requirements
Inferences
Load
View
Search
Mark Redundant Requirements
Domains
Projects
Products
Parts
Liaisons
Constraints
Maintenance
Standards
Product Status
Delivery Method
Diagrams
Modifications
Training

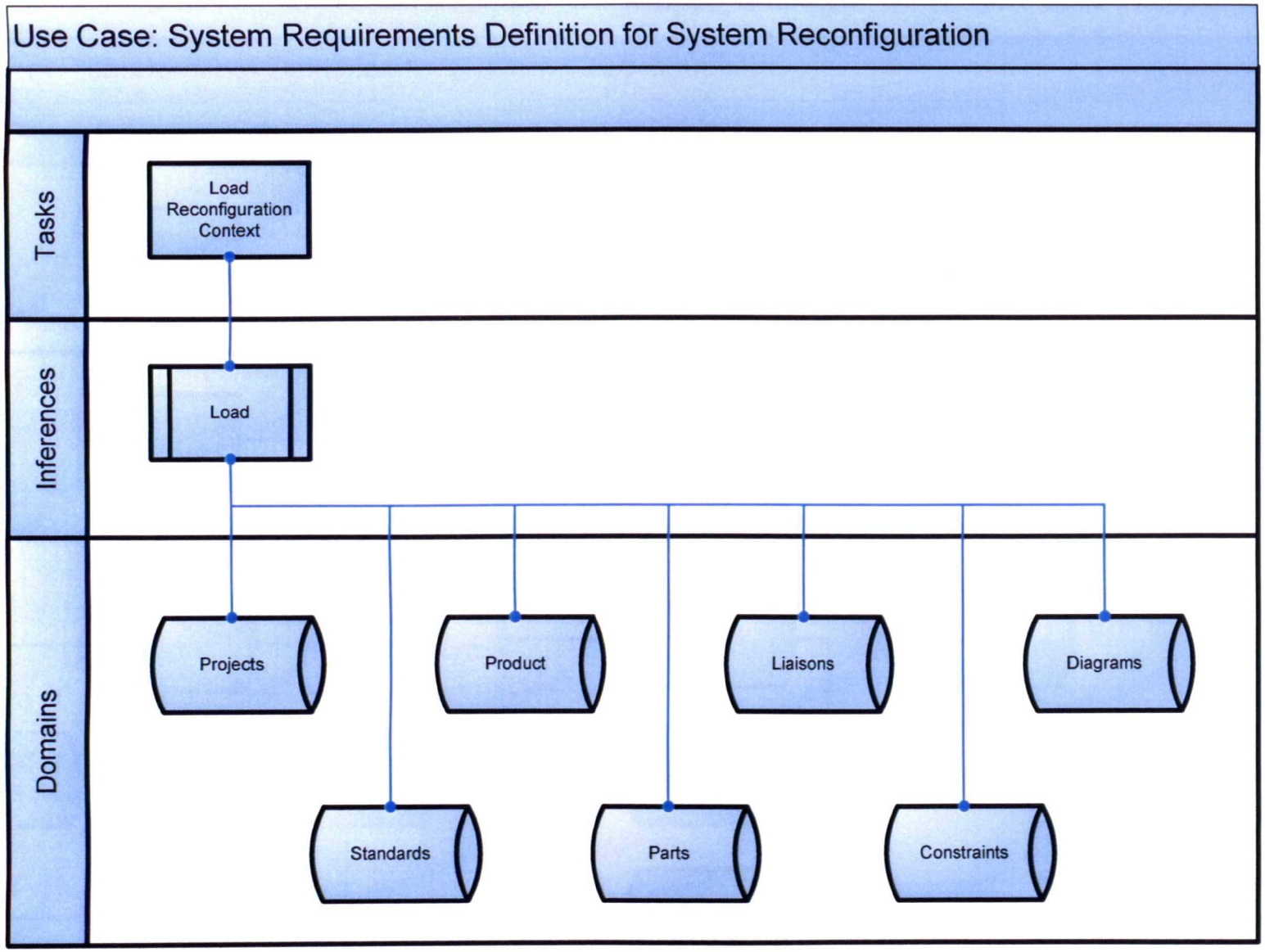

Use Case: System Requirements Definition
Tasks
Inferences
Domains
Map User Requirements to System Requirements
Load
Map – Choose Operations
Map – Choose Feeding
Map – Choose Material Transfer System
Map – Choose Reconfiguration Level
Projects
Products
Modifications
Operations
Feeding
Material Transfer
Standards
Product Status
Maintenance
Effectors
Level of Reconfiguration
Parts
Liaisons
Constraints
Process Elements
Control Architecture
Operational Requirements
Delivery Method
Diagrams
Training
Actions

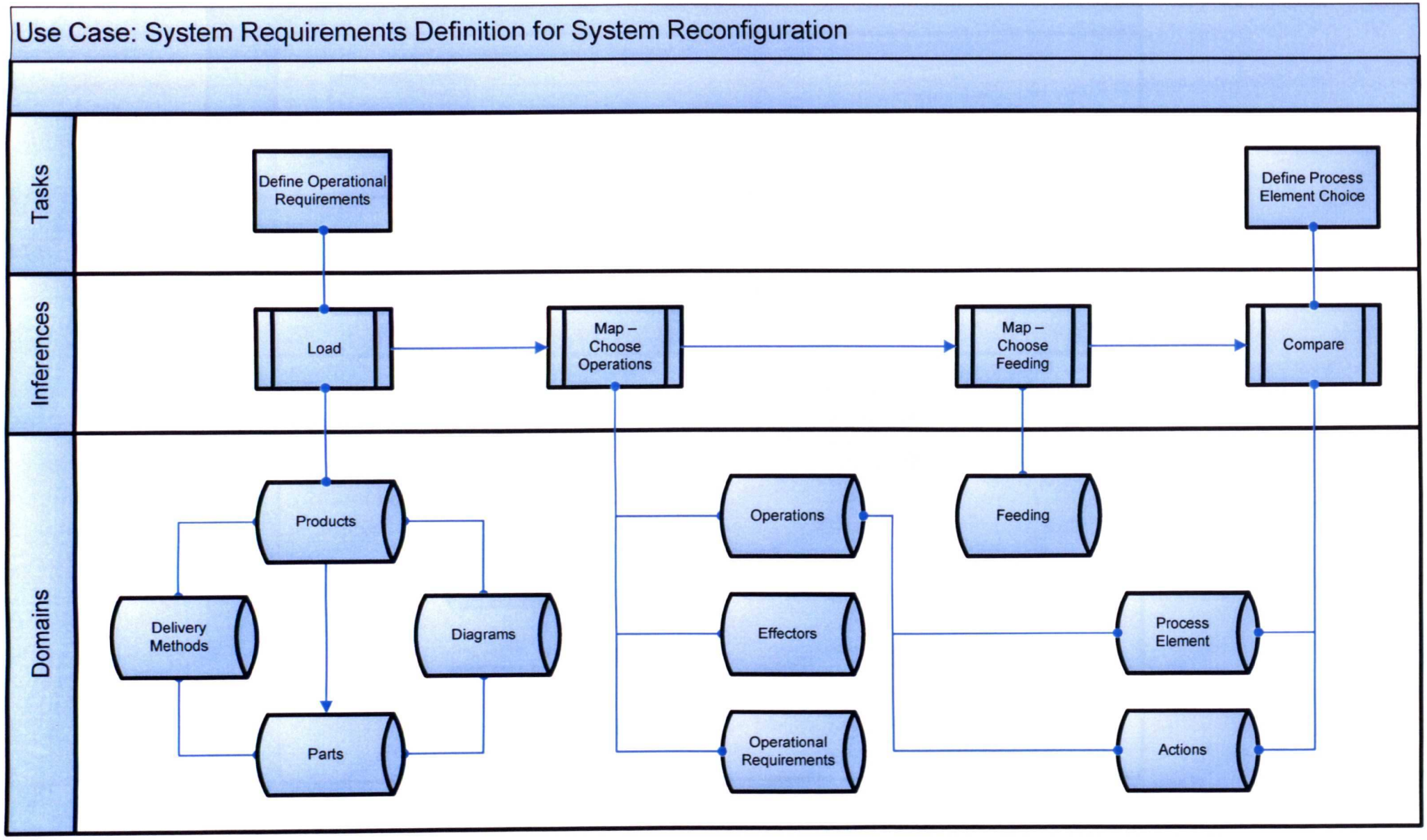

Use Case: System Requirements Definition
Tasks
Inferences
Domains
Formalise System Requirements
Confirm
Save
Send
System Overview
Tasks
Material Transfer
Control Architecture
Operational Requirements
Level Of Modularisation
Operations
Actions
Process Elements
Feeding
Effectors

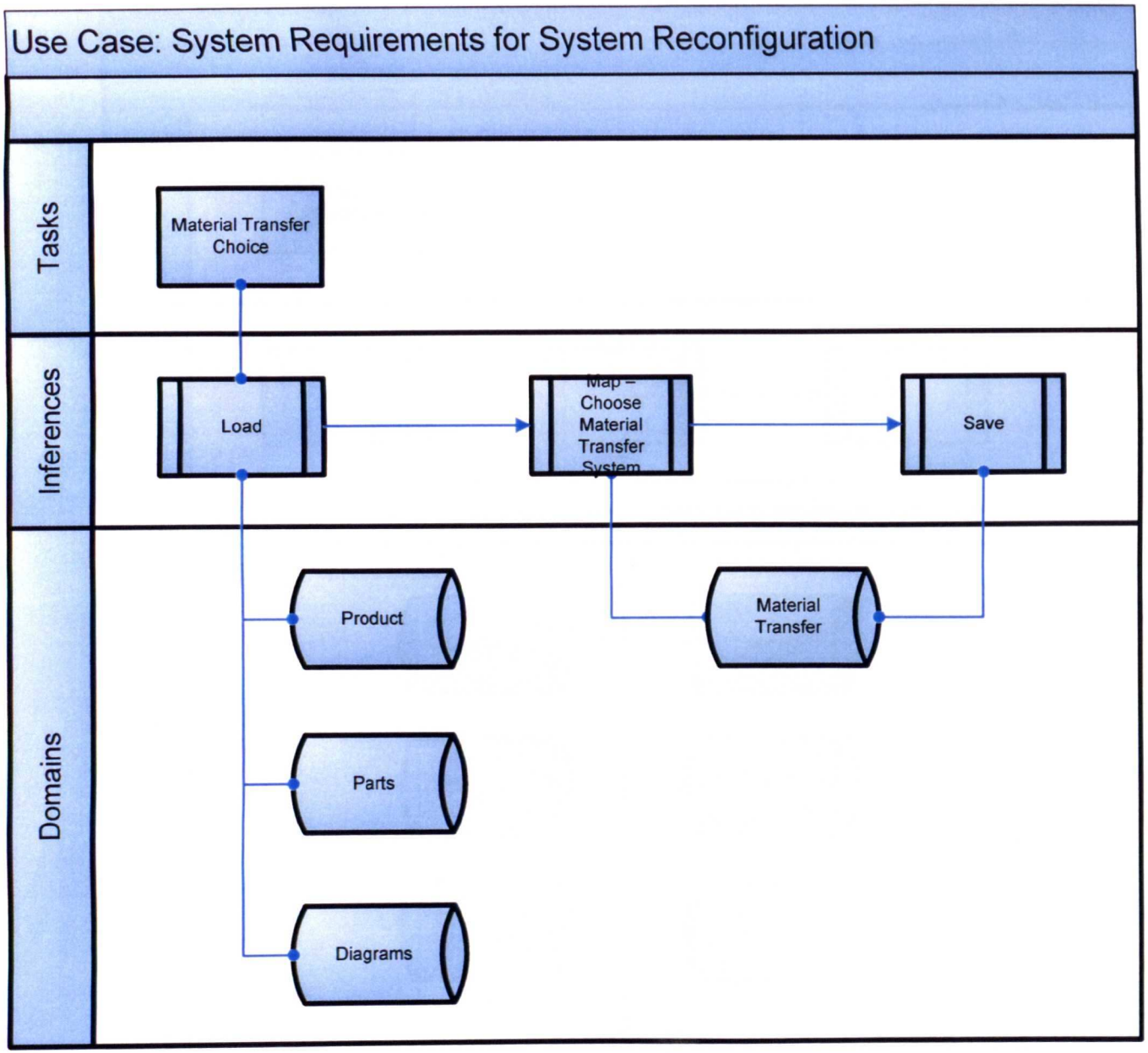

Use Case: User Requirements Definition for System Reconfiguration
Tasks
Inferences
Domains
Specify Business Requirements
Pointer to Domain
Enter Data
Save
Send
Projects
Standards

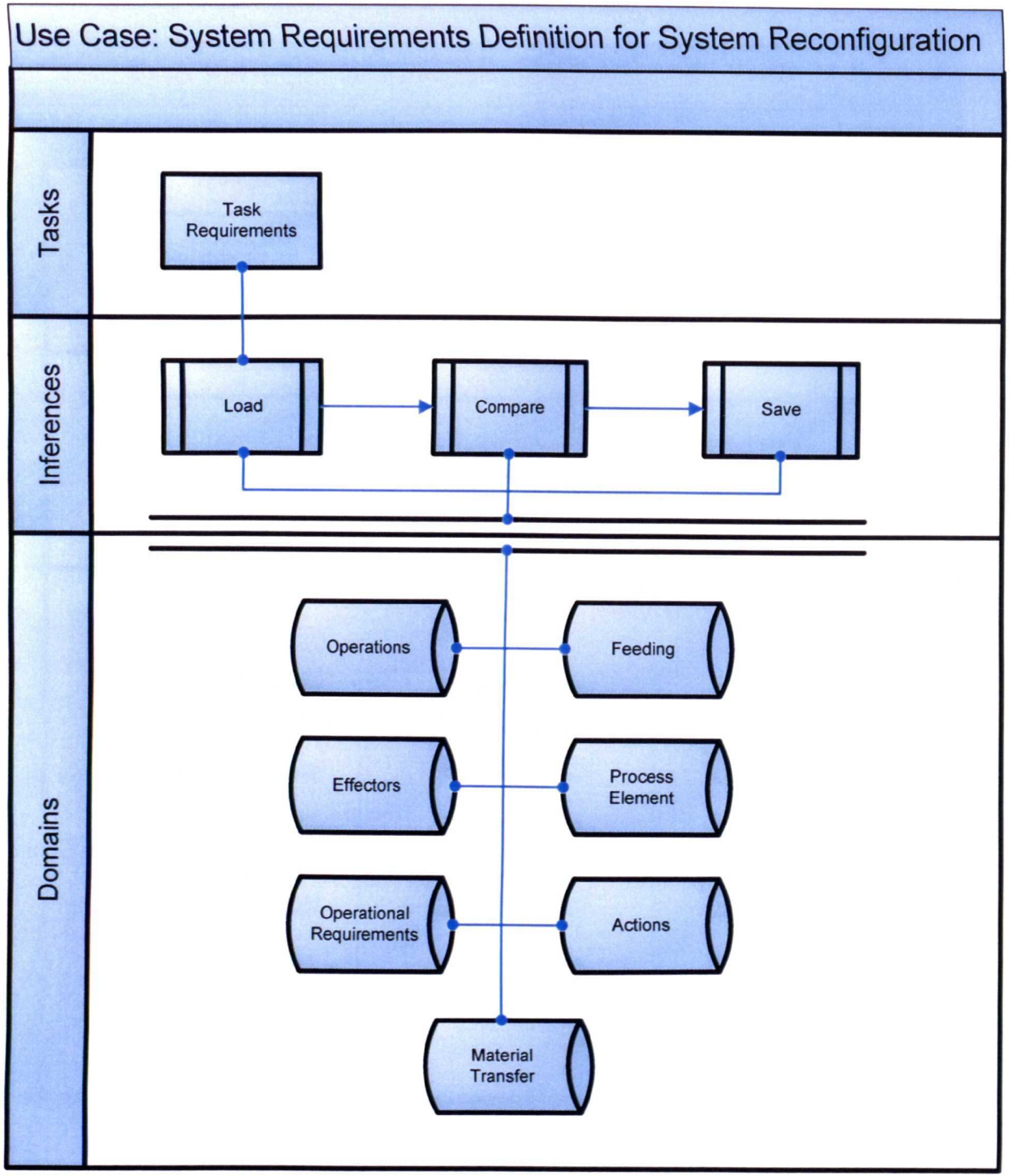

Use Case: User Requirements for System Reconfiguration
Tasks
Inferences
Domains
Specify Part Changes
Pointer to Domain
Load
Enter Data
Save
Send
Product
Parts
Diagrams

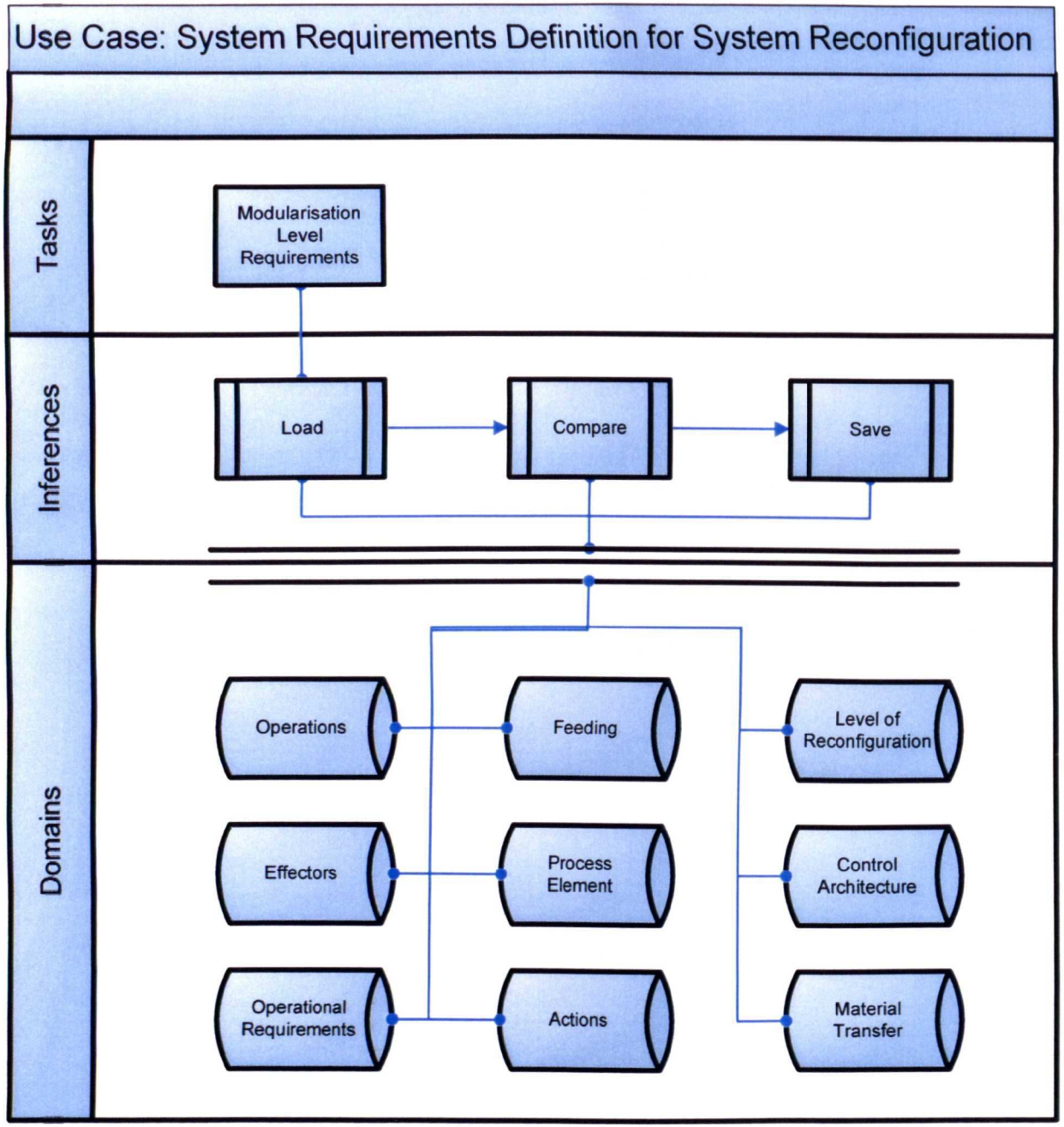

Use Case: User Requirements Definition for System Reconfiguration
Tasks
Specify Liaison Changes
Inferences
Pointer to Domain
Load
Enter Data
Save
Send
Domains
Parts
Liaisons
Constraints


Use Case: System Requirements Definition for System Reconfiguration
Tasks
Load Reconfiguration Context
Inferences
Load
Domains
Projects
Product
Liaisons
Diagrams
Standards
Parts
Constraints


Use Case: System Requirements Definition for System Reconfiguration
Tasks
Inferences
Domains
Define Operational Requirements
Define Process Element Choice
Load
Map – Choose Operations
Map – Choose Feeding
Compare
Products
Operations
Feeding
Delivery Methods
Diagrams
Effectors
Process Element
Parts
Operational Requirements
Actions


Use Case: System Requirements for System Reconfiguration
Tasks
Material Transfer Choice
Inferences
Load
Map – Choose Material Transfer System
Save
Domains
Product
Material Transfer
Parts
Diagrams


Use Case: System Requirements Definition for System Reconfiguration
Tasks
Task Requirements
Inferences
Load
Compare
Save
Domains
Operations
Feeding
Effectors
Process Element
Operational Requirements
Actions
Material Transfer


Use Case: System Requirements Definition for System Reconfiguration
Tasks
Modularisation Level Requirements
Inferences
Load
Compare
Save
Domains
Operations
Feeding
Level of Reconfiguration
Effectors
Process Element
Control Architecture
Operational Requirements
Actions
Material Transfer